Talent

Computador Personal DPC-200

Manual de uso y MSX Basic

Talent MSX

La idea del mundo en computación!

COMPUTADOR PERSONAL DPC-200

MANUAL DE USO Y MSX BASIC

TELEMATICA S.A.

Introducción

Bienvenidos al mundo de **MSX** a través de su computadora **TALENT MSX DPC 200**

El sistema **MSX** fue concebido por la firma **ASCII-Microsoft** para que definitivamente se ordene el mundo de las computadoras personales y su idioma más popular: el **BASIC.**

Dicho orden surge a través de esta norma materializado en lo que en computación llamamos **compatibilidad.**

Este concepto, que hasta ahora nunca se aplicó, siempre trajo aparejado el siguiente problema (Un ejemplo entre varios...):

Amigo 1:- ¡Tengo el juego **Los monstruos de Klingon** que es buenísimo!
Amigo 2:- ¿Podré conseguir una copia?
Amigo 1:- Si pero... no te va a servir, ya que mi máquina es la XX-1000 y vos tenés una WW-32 .
Amigo 2:- ¡±#$%$#±!!!

Gracias a la **Norma MSX,** Ud. puede adquirir cualquier computadora que tenga el sello **MSX,** y podrá utilizar todo los programas disponibles con dicho sello, utilizar los programas desarrollados en otras **MSX** en su —TALENT MSX — . Asimismo, si Ud. desarrolla programas en su —TALENT MSX — , cualquier usuario de otras marcas con sistema **MSX,** podrá ejecutarlos sin ningún inconveniente.

Además su – TALENT MSX – posee otras ventajas: se aprovechó al máximo la norma **MSX** en cuanto a las capacidades permitidas de memoria, incorporándosele **64 Kbytes** de memoria RAM, con lo cual Ud. podrá utilizar todos los programas desarrollados en **CP/M** para esta configuración. O sea, su – TALENT MSX — puede trabajar en MBASIC, PASCAL, FORTRAN, COBOL, C, utilizar los famosos dBASE II, Multiplan, etc.

También los archivos generados en el sistema operativo **MS-DOS** pueden ser leídos por su — TALENT MSX — Esto significa que con su "humilde servidora electrónica" Ud. puede trabajar como si se tratara de un verdadero sistema profesional que hasta hace muy poco tiempo sólo podían disponer las empresas o laboratorios especializados.

Finalmente, para el usuario de computadoras personales, decimos que el **BASIC MSX** es un desarrollo de **MS-BASIC versión 4.5** con muchas instrucciones tomadas del **GW-BASIC.**

Pero **MSX** significa **Microsoft Extended BASIC**, o sea que no es "sólo" lo dicho anteriormente, si no que se le han agregado las siguientes ventajas acordes a la capacidad de su máquina, como ser:

(1) Gráficos. Manejo de **SPRITES**, y el micro lenguaje de gráficos utilizable mediante la instrucción **DRAW**; que no es ni más ni menos que la característica gráfica del lenguaje **LOGO.**

(2) Sonido. Tiene la capacidad de generar música a tres voces con efectos especiales utilizando las instrucciones **PLAY** y **SOUND.**

(3) Interrupciones. Cada 1/5Ø de segundo, Ud. puede chequear si se produjeron o no determinados **eventos** programables. Esto es algo único de **MSX**, y permite realizar programas en **BASIC** con tiempos de reacción dignos de código de maquina

Lea atentamente este manual y tómese su tiempo para familiarizarse con su nueva computadora personal.

Este manual describe las operaciones básicas para utilizar la **TALENT DPC 200** correctamente, su puesta en marcha y las instrucciones incorporadas en el **MSX BASIC.** La descripción de cada instrucción viene acompañada con ejemplos de programas ad-hoc.

AVISO:

1. Telemática S.A. puede cambiar el contenido de este manual sin previo aviso.
2. Cuando se utiliza un programa o cálculo especial en el Talent MSX DPC 200 compruebe la secuencia de ejecución, resultados intermedios y los finales
3. Telemática S.A. no se hace responsable de ninguna pérdida financiera del usuario como consecuencia del uso de la computadora Talent MSX DPC 200.

* **MSX** y **MS-DOS** son marcas registradas de MICROSOFT CORPORATION.
* **dBASE II** es una marca registrada de ASHTON TATE.

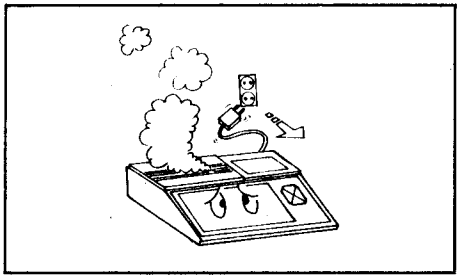
* **CP/M** es una marca registrada de DIGITAL RESEARCH.

(C) 1985 Telemática S.A.

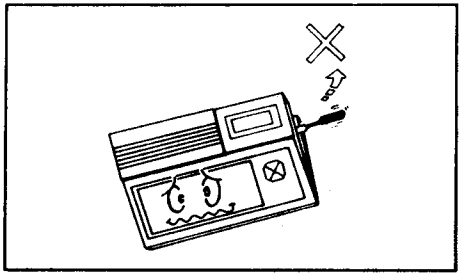
Precauciones en el uso de su TALENT MSX DPC 200

Los siguientes son útiles consejos para el uso de su TALENT MSX DPC 200:

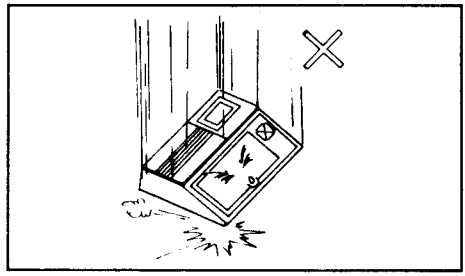
Si algo va mal durante el uso de su Talent MSX, y ninguna de las indicaciones del Apéndice J surte efecto, desconéctela inmediatamente. No es recomendable que la siga utilizando en estas condiciones. Póngase en contacto con su distribuidor Talent tan pronto como sea posible.



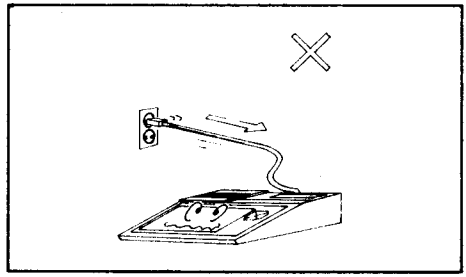
No desarme ni modifique su Talent MSX DPC 200



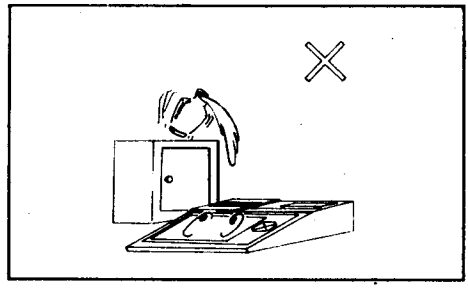
No permita que su Talent MSX DPC 200 esté sometida a golpes o maltratos de ningún tipo.



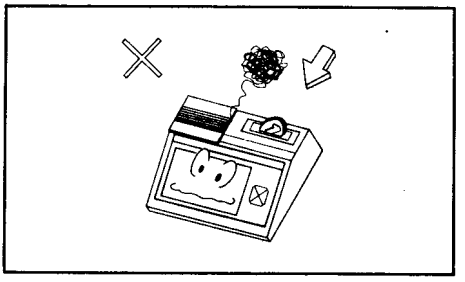
No maltrate los cables. Cuando desconecta su computadora, asegúrese de tomar el cable por el enchufe y recién tire de él.



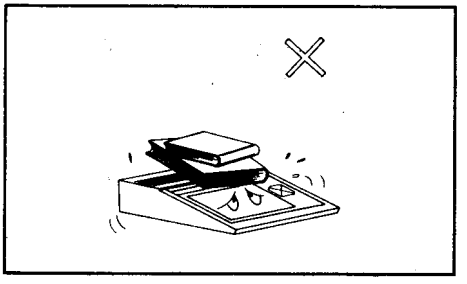
Evite usar o dejar su Talent MSX DPC 200 en un lugar expuesto a la humedad.



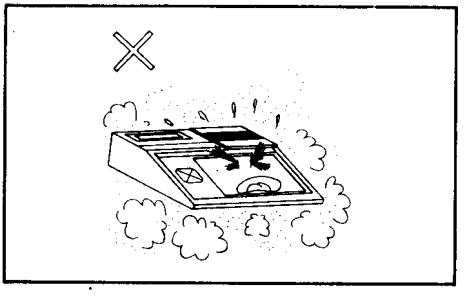
No introduzca ningún material extraño, en especial agujas, monedas ú objetos inflamables en el zócalo para inserción de cartuchos ROM.



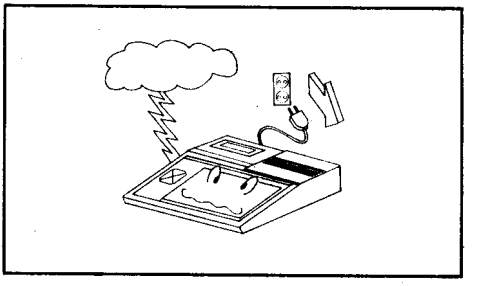
No coloque nada encima de su Talent MSX DPC 200.



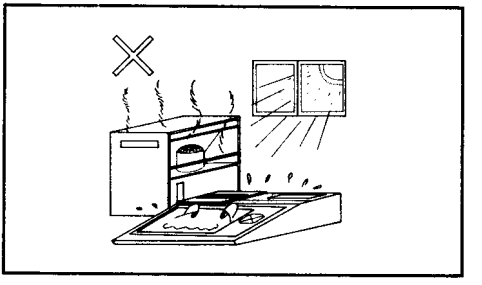
Evite usar o dejar su Talent MSX DPC 200 en lugares de extrema humedad o muy polvorientos.



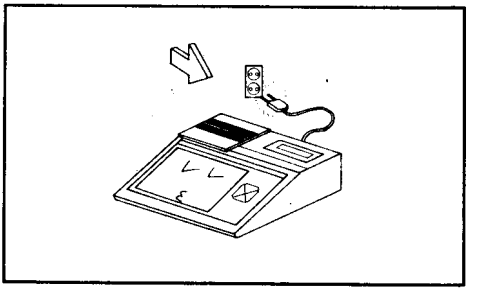
Si escucha muchos truenos durante una tormenta, desconecte su Talent MSX DPC 200 de la entrada de 220V.



Evite usar o dejar su Talent MSX DPC 200 en un lugar expuesto directamente a los rayos solares y no la ponga cerca de ninguna fuente de calor.



Cuando no utilice su Talent MSX DPC 200 por un largo período de tiempo, desconéctela de la entrada de 220V.



Limpieza.

Es recomendable limpiar su Talent MSX DPC 200 periódicamente para prolongar su vida útil. Tenga en cuenta los siguientes puntos cuando limpie su computadora.

* Utilice un paño suave para limpiar el gabinete.

* Cuando la Talent MSX DPC 200 está extremadamente sucia, aplique detergente neutral diluído con agua a un paño y limpie su computadora con él; luego séquela con un paño seco.

* Evite usar paños químicamente tratados y no utilice alcohol, thinner, u otros solventes. Si se usa estos solventes o se deja encima de la computadora algún objeto hecho de goma o vinilo se dañara la superficie del gabinete de su Talent MSX DPC 200.

Muy Importante

GARANTIA

Telemática S.A. garantiza por 6 meses el buen funcionamiento de este equipo, a partir de la fecha de compra, comprobada por la factura correspondiente. Esta garantía sólo cubre el uso del equipo dentro de las especificaciones y no cubre los abusos, maltratos, accidentes y/o soporte lógico (programas). Telemática S.A. se limitará a reparar o reemplazar el equipo, a su voluntad, para restituir su funcionamiento normal, sin ningún otro compromiso.
Este equipo no tiene atención a domicilio, sólo en los Servicios Técnicos Autorizados.
Esta garantía cesará si personas ajenas al Servicio Técnico Autorizado trataran de ajustar o reparar el equipo.

Service Central de Telemática S.A.: Chile 1347 (1098) Capital
Tel: 37-0051/54

Horario: 9:30 a 12:30 hs. y 13:30 a 17:00 hs

Instalación del sistema

Instalación del sistema

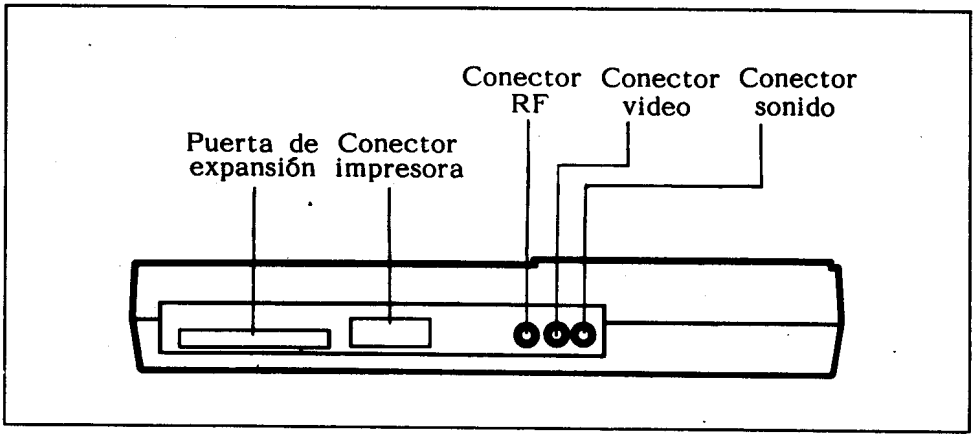
Contenido del embalaje del Talent MSX DPC 200.

Una vez desempaquetada su Talent MSX DPC 200 y antes de conectar la computadora, asegúrese que junto con este manual han sido suministrados los siguientes accesorios:

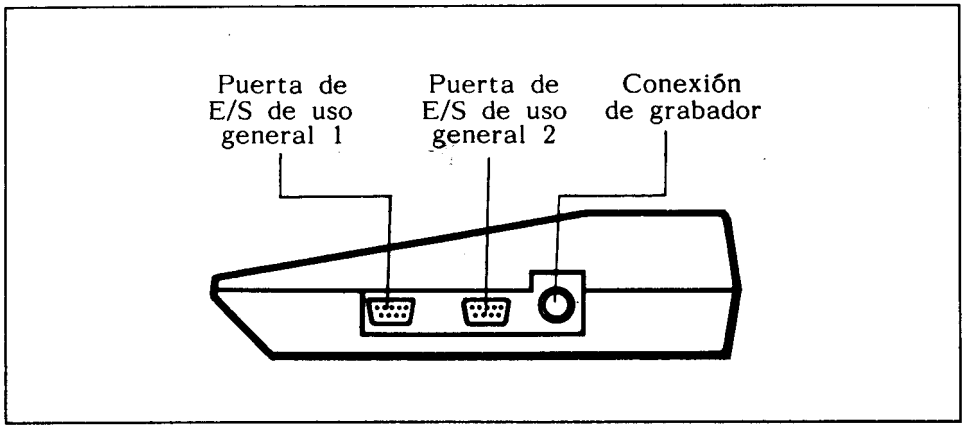
Teclado (la computadora misma)
Cable de grabador
Cable de conexión a TV

Si alguno de los elementos antes citados no ha sido suministrado, por favor consulte con su distribuidor.
Conserve la caja para cuando necesite trasladar su computadora.

Conexiones e interruptores.



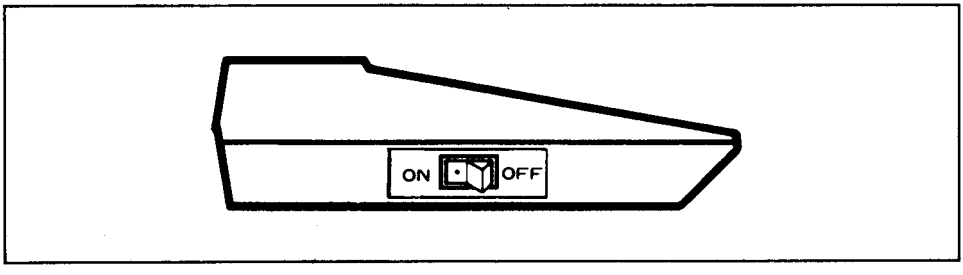
Puerta de expansión: Conecta un periférico externo a la computadora.
Conector impresora : Conecta la impresora a la computadora.
Conector video: Conecta su computadora a un monitor o a la entrada de video de su T.V. (si la tiene)
Conector sonido: Conecta su computadora a un monitor o a la entrada de audio de su T.V. (si la tiene).
Conector RF: Conecta su computadora a la entrada de antena de su T.V.



Conexión de grabador: Conecta la computadora al grabador de cassette

Puerta de E/S de uso general:

Esta puerta conecta los joystick, tableta gráfica o paddle a su computadora (todos vendidos por separado).



Interruptor: Enciende su computadora.

Nota: Utilice únicamente accesorios recomendados por Telemática S.A. en su computadora Talent MSX DPC 200.

PUESTA EN MARCHA

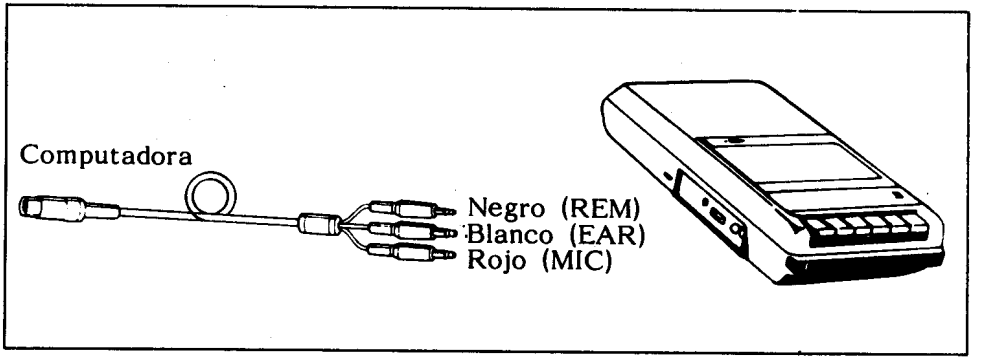
1. Conexión al televisor.

Con la computadora y el televisor apagados, el cable de T.V. suministrado debe ser conectado entre el conector de RF en la parte de atrás de la computadora y la antena de su televisor. Cabe aclarar

que el cable suministrado es de 75 ohms de impedancia y si su antena es de 300 ohms, deberá utilizar un adaptador.

2. Conexión al grabador de cassette.

Para conectar a un grabador de cassette, el cable de ser puesto entre el conector de grabador de la computadora y los conectores de entrada, salida y mando remoto del grabador del modo siguiente:



Conecte el enchufe de 8 pins DIN en el lado correspondiente de la computadora.
Conecte el grabador en este orden:

Conector blanco	A la salida EARPHONE o MONITOR del grabador.
Conector rojo	A la entrada MIC o EXT MIC del grabador.
Conector negro	A la entrada de REMOTE del cassette.

Nota: algunos grabadores tienen invertida la polaridad del REMOTE. Si es su caso, la cinta no avanzará cuando pulse PLAY y este encendido el motor desde programa (MOTOR ON). Desconecte este conector para solucionar el problema y controle manualmente el grabador.

3. Sintonía del televisor a su computadora.

Conectar el televisor y su computadora a un tomacorriente 220V. Enciénda la computadora con su interruptor y el televisor.
El indicador de tensión del teclado de la Talent MSX DPC 200 se debe iluminar. Si no, apague la computadora y compruebe el cableado y el fusible. Si el indicador de tensión continúa apagado consulte el Apéndice J.

Con el televisor y su computadora encendidos, seleccione el canal más conveniente (3 ó 4) y ajuste el control de sintonía para obtener el siguiente mensaje en la parte central de la pantalla:

```
MSX System
Version 1.0
Copyright 1983 by Microsoft.
```

Luego de unos segundos, aparece el siguiente mensaje:

```
MSX BASIC Version 1.0
Copyright 1983 by Microsoft.
28815 Bytes free
OK
```

Su sistema está listo para funcionar.

La computadora esta diseñada para proveer una protección razonable contra interferencias en sus equipos de recepción. Sin embargo, no se garantiza que esta interferencia no ocurrirá en la recepción de radio o televisión, que comenzará desde el momento en que enciende su computadora. Para corregir este problema, le sugerimos lo siguiente:

* Reoriente la antena receptora.
* Reubique la computadora respecto del receptor.
* Aleje la computadora del receptor.
* Enchufe la computadora a un tomacorriente distinto al del receptor para que queden conectados a distintas ramas del circuito eléctrico de su casa.

De ser necesario, le sugerimos que contacte a su distribuidor o un técnico experimentado de radio/televisión para sugerencias adicionales.

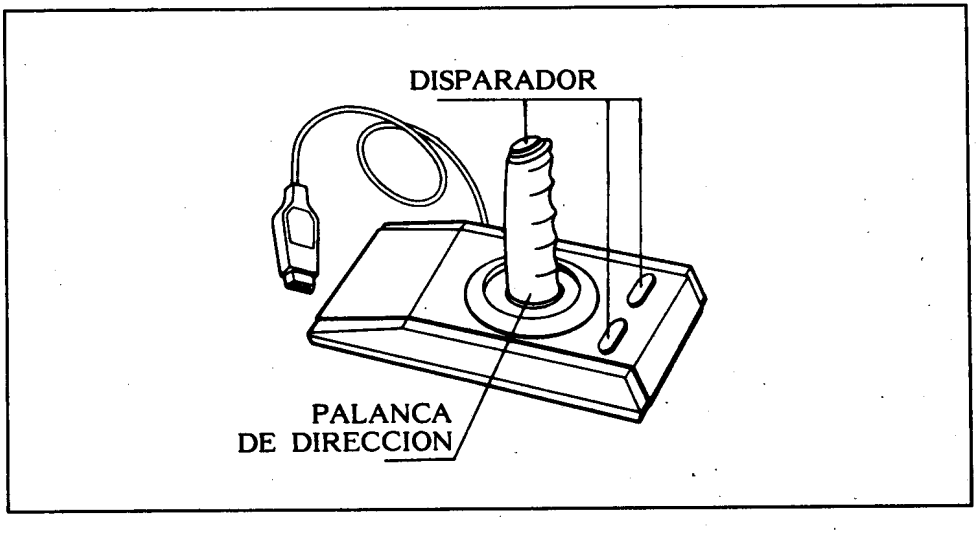
4. Conexión a un sistema de audio.

Para conectar a un sistema de audio (por ejemplo estéreo o HI-FI) el cable debe ser conectado entre el conector EXTERNAL IN del sistema de audio y el conector de audio de la computadora.

Para obtener sonido, digite BEEP en el teclado de la computadora y luego la tecla [RETURN]. Si el sistema de audio tiene selector, éste debe ser colocado en posición MONO. Donde no haya selector, el sonido saldrá a través de un solo parlante

5. Expansión de las capacidades de su Talent MSX DPC 200.

Su computador personal Talent MSX DPC 200 es completamente expandible, permitiéndole añadir por ejemplo unidades de disco para almacenamiento de datos, una impresora para listar programas o escribir cartas y dos joystick para poder jugar. Por ejemplo, si desea conectar un joytick, enchufe el conector de 9 pins en el costado de su computadorā:



Consulte a su distribuidor de Telemática para más detalles de estos periféricos.

6. Carga de un programa desde una cinta de cassette.

Después de haber conectado el grabador adecuado, siguiendo el método señalado en "conexión a un grabador", se puede dar tensión a la computadora y al televisor, introduciéndose el cassette elegido en el grabador.

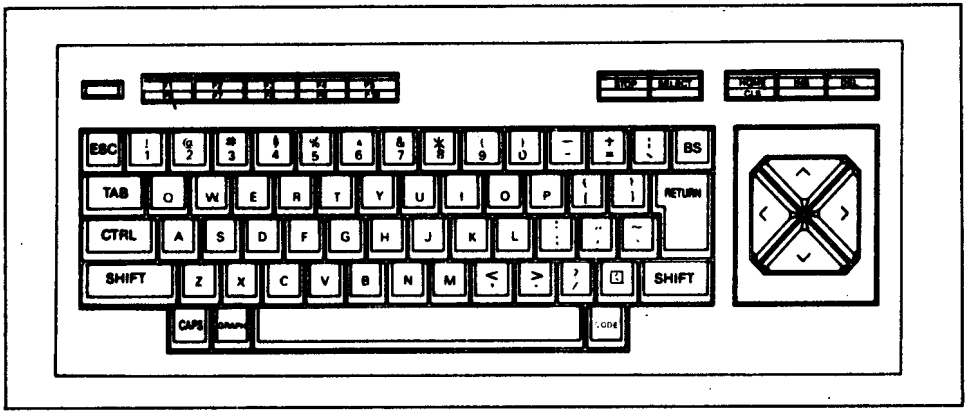
Teclado

Teclado

La comunicación con su Talent MSX DPC 200 se realiza generalmente enviándole instrucciones a traves del teclado. Sus instrucciones y lo que responde la computadora puede visualizarse en la pantalla del televisor que está conectado a su computadora.

El teclado de su computadora le debe resultar bastante familiar, ya que se asemeja bastante a una máquina de escribir. Sin embargo, el teclado contiene teclas adicionales que son necesarias para la comunicación efectiva con su Talent MSX DPC 200.

Esquema del teclado



El teclado de su computadora consiste en teclas alfabéticas, teclas numéricas, teclas de caracteres especiales y de funciones especiales.

Teclas de control

STOP

Si presiona esta tecla, se hará una pausa en la ejecución del programa.
Si la presiona nuevamente, se continuará con la ejecución del mismo.

CTRL

Es la tecla de CONTROL. Presionando esta tecla y una tecla de caracter simultáneamente le indica a la computadora que ejecute una función especial.
(Vea el Apéndice A para más detalles)
Presionando esta tecla y la tecla STOP simultáneamente hará finalizar la ejecución del programa y le devolverá el control a usted.

RETURN

Pulse esta tecla al final de cada instrucción que tipee. Al presionar esta tecla le indica a su computadora que ingrese la instrucción que usted ha digitado en su memoria.
Nota: RETURN le indica que presione la tecla "RETURN"

SELECT

Esta tecla no tiene función en programación BASIC y solamente tienen acceso a ella los programas. Si presiona esta tecla, se genera el código de control 24.

ESC

Esta tecla no tiene función en programación BASIC y solamente tienen acceso a ella los programas. Si presiona esta tecla, se genera el código de control 27.

Teclas de edición

INS

Esta tecla se utiliza cuando usted desea insertar caracteres dentro de una línea.
Funciona como un conmutador del modo de inserción. Cuando el modo de inserción está activo, el tamaño del cursor se reduce y los caracteres se insertan en la posición del cursor. Los caracteres a la derecha del cursor se mueven hacia la derecha a medida que se insertan caracteres. Se mantiene el justificado de líneas. Cuando se desactiva el modo de inserción, el cursor vuelve al tamaño normal.

DEL

Esta tecla se utiliza para borrar el caracter donde está ubicado el cursor. Todos los caracteres a la derecha del cursor se corren un espacio hacia la izquierda. Se mantiene el justificado de líneas.

BS

BACK SPACE (retroceso). Borra el caracter ubicado a la izquierda del cursor.

Todos los caracteres a la derecha del cursor se corren un espacio hacia la izquierda. Se mantiene el justificado de líneas.

TAB

Esta tecla mueve el cursor hasta la siguiente tabulación escribiendo espacios. Las tabulaciones están ubicadas cada 8 caracteres.

HOME

Esta tecla mueve el cursor a la esquina superior izquierda (posición de origen (home)). Si presiona esta tecla y la tecla SHIFT simultáneamente, además de lo anterior se borra la pantalla.

CAPS

Si presiona esta tecla se conmutará la impresión de teclas alfabéticas de minúsculas a mayúsculas y viceversa.

SHIFT

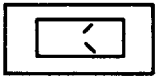
Digitando teclas alfabéticas, funciona como la tecla CAPS.
Si presiona esta tecla y un número o un caracter especial se imprimirá el símbolo impreso en la parte superior de esa tecla.

GRAPH

Si presiona esta tecla y una tecla de caracter simultámente se imprime un símbolo gráfico.
El diseño detallado de cada símbolo se muestra más adelante.

CODE

Presione esta tecla y una tecla de caracter simultáneamente y obtendrá los caracteres de símbolos especiales.
El diseño detallado de cada símbolo se muestra más adelante.



Tecla muerta. Si presiona esta tecla y luego una vocal, se imprime la vocal acentuada. Con **SHIFT** obtiene la acentuación en castellano.

Teclas de control del cursor

Las teclas de control del cursor (arriba, abajo, izquierda y derecha) controla los movimientos del cursor.
Si presiona, por ejemplo, las flechas arriba e izquierda simultáneamente, el cursor se mueve en diagonal hacia la esquina superior izquierda de la pantalla.

Otras combinaciones de este tipo funcionan igual, con lo cual usted puede mover el cursor en 8 direcciones distintas utilizando estas teclas.

Teclas de funciones

Estas teclas están ubicadas en la fila superior izquierda del teclado, y cada una está indicada con la letra "F". Son "dispositivos para ahorrar trabajo", ya que le permiten indicarle a su computadora que ejecute las funciones más utilizadas presionando sólo una tecla en vez de tener que tipear muchas teclas.

Aquí se hace una lista de las funciones que están asignadas a cada tecla cuando enciende su computadora.

Las teclas de función F1 a F5 se operan presionando la tecla adecuada. Para acceder a las funciones de F6 a F10, pulse la tecla SHIFT y luego recién pulse la tecla de función deseada.

Listado de las funciones iniciales de las teclas de función.

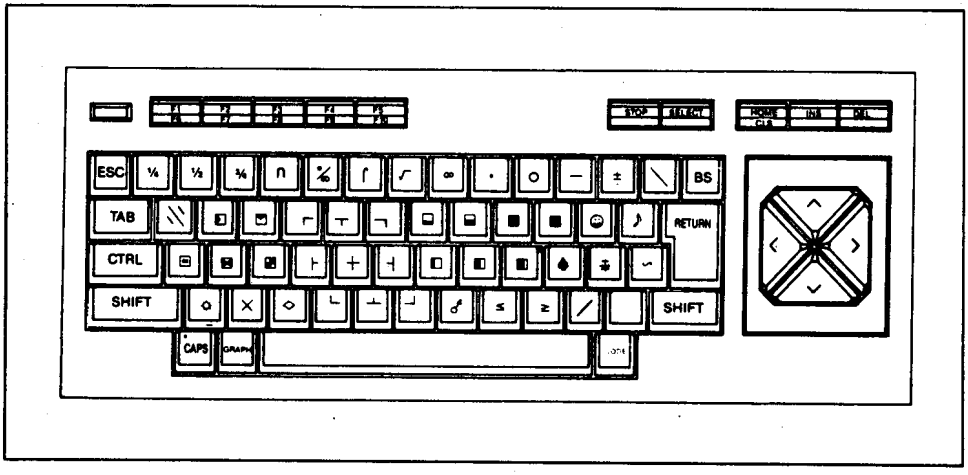
F1 color	El comando color se utiliza para cambiar el color del texto, de la pantalla y de los bordes de la misma.
F2 auto	El comando auto se utiliza para hacer que la computadora genere automáticamente las líneas del programa. Este comando se utiliza muy a menudo, ya que todas las líneas deben estar precedidas por su número de línea.
F3 goto	goto es un comando que le permite ejecutar un programa desde la línea que desee sin alterar el contenido de las variables.

F4	list	Este comando le permite listar el programa almacenado en memoria.
F5	run+ return	run le indica a la computadora que ejecute el programa que usted escribió.
F6	color 15,4,4+ return	Este comando le indica a su computadora que imprima letras blancas sobre fondo azul y con borde del mismo color. Estos son los colores de la pantalla cuando enciende su computadora.
F7	cload"	cload" le indica a su computadora que ingrese (carge) datos desde un grabador de cassette (que se conecta fácilmente a la misma)
F8	cont+ return	Con este comando se puede "continuar" la ejecución del programa luego de la última línea ejecutada.
F9	list.+ return	Con LIST. (con un punto cerca de la instrucción) únicamente se imprime la última línea que usted estuvo trabajando (ya sea programando, editando, etc.)
F10	CLS+ run+ return	Este comando es igual al RUN con el agregado de que ahora se borra también la pantalla antes de que se "corra" el programa.

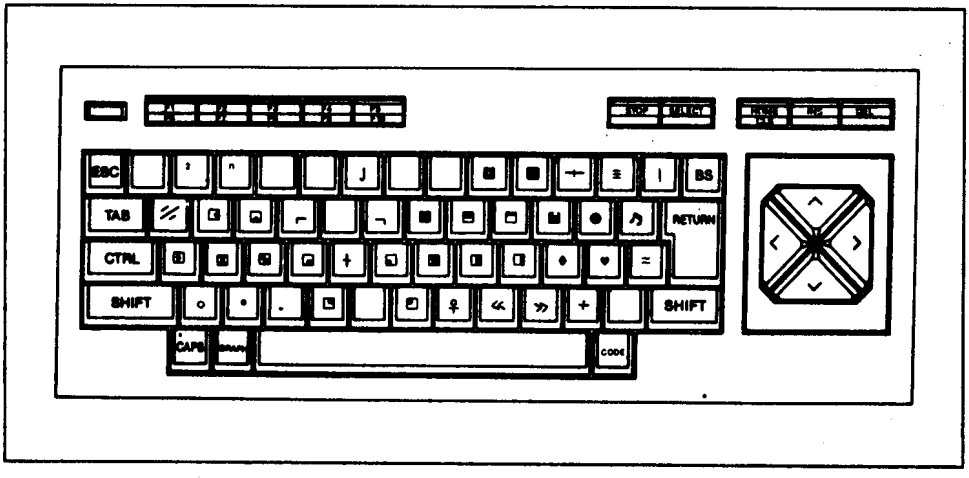
La computadora muestra normalmente en pantalla las funciones asignadas a las teclas F1 a F5 y cada vez que presione la tecla shift se imprimen las funciones de las teclas F6 a F10.

Cualquiera de estas funciones predefinidas pueden cambiarse fácilmente a su conveniencia. Lea el comando "KEY" en la sección de MSX BASIC para más detalles.

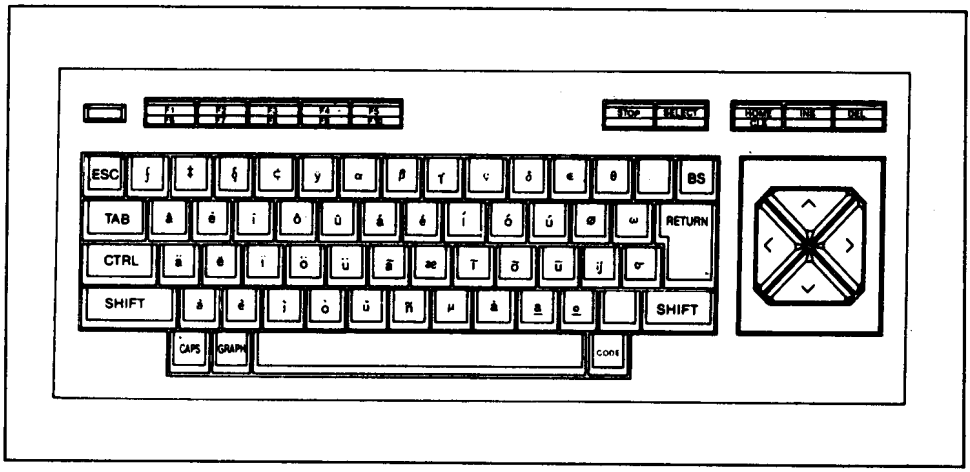
ESQUEMA DE LAS TECLAS ACTIVADAS CON GRAPHIC.



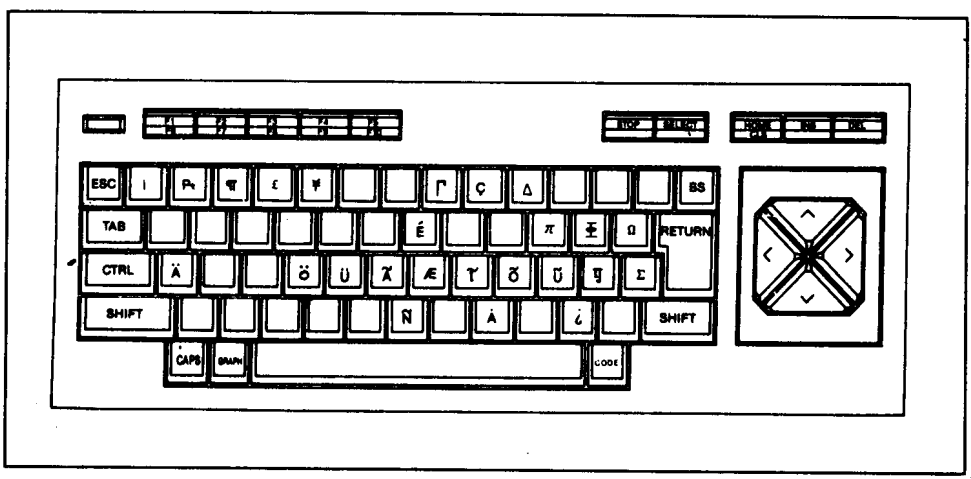
ESQUEMA DE LAS TECLAS ACTIVADAS CON GRAPHIC+SHIFT.



ESQUEMA DE LAS TECLAS ACTIVADAS CON CODE.



ESQUEMA DE LAS TECLAS ACTIVADAS CON CODE+SHIFT.



Programación MSX BASIC

Programación MSX BASIC

I - INTRODUCCION

1. QUE ES EL MSX BASIC?

BASIC es el lenguaje por intermedio del cual usted puede comunicarse con su computadora. Fue creado en 1965 en el Dartmouth College por dos maestros: Thomas Kurtz y John Kemeny. La idea original fue la de tener un medio sencillo con el cual programar su computadora.

Sin embargo, con la introducción de la computadora personal, el uso del BASIC se ha hecho prácticamente universal y esto ha supuesto que en el mismo se han introducido ligeras variaciones en el lenguaje.

Estas variaciones son las que hacen incompatibles las diferentes computadoras, pero gracias a la norma MSX, 21 firmas (hasta el momento) producen computadoras con esta norma, y tienen la gran ventaja de poder utilizar los programas desarrollados en una marca en cualquier otra, con tal de que sea una computadora MSX.

2. LENGUAJE BASIC

El lenguaje BASIC consiste en comandos, sentencias, y funciones. Sin embargo, no hay una diferenciación precisa entre comandos y sentencias.

* **Comandos** se utilizan para controlar el ingreso, ejecución y manejo del programa, mayormente en modo Directo.

* **Sentencias** son los elementos básicos de un programa BASIC, y se utilizan para indicar a la computadora qué debe realizar. Las sentencias de un programa deben estar precedidas por su correspondiente número de línea.

* **Funciones** permiten ejecutar operaciones aritméticas o manipular caracteres numéricos y alfabéticos (llamados datos alfanuméricos). Se utilizan en las sentencias.

3. Programas BASIC

Un programa BASIC es un conjunto de sentencias y/o comandos ubicados en el orden correcto para resolver problemas específicos o ejecutar ciertas tareas. Su computadora ejecuta sentencias y comandos en, como regla, orden ascendente de número de línea asignados a cada línea.

(1) Línea de programa

Cada línea de programa consiste en un número de línea y una o más sentencias o comandos. Es el elemento mínimo utilizado en un programa BASIC.

nnnn sentencia

(nnnn = número de línea)

* Hasta 255 caracteres, incluyendo los números de línea, pueden utilizarse para escribir una línea de programa.
* El número de línea es un número entero en el rango de 0 a 65529.
* Cada línea puede tener varios renglones, siendo un renglón el total de caracteres que se puede imprimir en el ancho de la pantalla.

Ejemplo:	número de línea	sentencia
	10	PRINT "COMPUTADORA"
	20	PRINT "TALENT MSX"

(2) Líneas multisentencia

Cada línea consiste, como regla, de una sentencia o comando. Si desea simplificar su programa, puede utilizar más de una sentencia o comando en una línea, separándolos con dos puntos (:)

nnnn sentencia: sentencia: sentencia

No es posible utilizar, sin embargo, líneas multisentencia a continuación de las siguientes órdenes:

REM, END, RETURN, GOTO

Ejemplo: 15 PRINT "TALENT": BEEP: PRINT "MSX": BEEP: GOTO 15

(3) MODOS DE EJECUCION

(1) **Modo Comando**

En el modo Comando, la computadora está lista para ejecutar un comando o sentencia entrado con el teclado, o para ingresar líneas de programa.

* Inmediatamente después de encender su computadora o finalizada la ejecución de un programa, la computadora entra en modo Comando con un "Ok" que aparece en pantalla.
* Si las teclas [CTRL] y [STOP] se aprietan simultáneamente durante la ejecución de un programa o de un comando, la computadora detiene la ejecución y entra en el modo Directo.

(2) **Modo Programa**
Si un comando o sentencia entrado a través del teclado está precedido de un número de línea, la computadora lo contemplará como una línea de programa. Para ejecutarlos, digite **RUN** o **GOTO** o **GOSUB**. Cuando ejecuta el programa en memoria está en modo Programa.

(3) **Modo Directo**
Si un comando o una sentencia es entrado sin número de línea la computadora lo ejecutará inmediatamente. Para detener la ejecución directa se deben apretar simultáneamente las teclas [CTRL] y [STOP], y la computadora entra en modo Comando.

II - DESARROLLO DE UN PROGRAMA BASIC

Secuencia de programación

(1) Entrada del programa
Entrar un programa deseado desde el teclado o el grabador de la computadora.

(2) Verificación del programa
Listar las líneas del programa en pantalla o mediante la impresora para comprobar su contenido.

(3) Corregir el programa

(4) Ejecutar el programa apretando la tecla **F5-F10**

(5) Si ocurre un error durante la ejecución o si el resultado es incorrecto, determinar el error usando el mensaje de error como guía. Después de corregir el error, repetir desde el paso (2).

(6) Grabar el programa en una cinta o dispositivo periférico.

(1) ENTRADA DEL PROGRAMA

1) Entrar las líneas del programa (cada una precedida de su número) mediante el teclado cuando la computadora está en modo Comando.
2) Pulsar la tecla [RETURN] al final de cada línea (si no se pulsa, la línea no quedará registrada en la computadora).

* Si la transferencia de datos desde o hacia un periférico está programada, especificar el número de archivos a utilizar, usando la sentencia **MAXFILES.**

* Para una mayor eficiencia en la entrada del programa usar los siguientes comandos BASIC:

AUTO: Permite a la computadora generar automáticamente números de líneas en orden secuencial. Para salir del modo "AUTO", apretar simultáneamente las teclas [CTRL] y [STOP].

KEY: Permite asignar funciones a las teclas de función.

KEY LIST: Lista las funciones asignadas a las teclas de función.

* Cuando se carga un programa de un periférico, utilizar los siguientes comandos:

CLOAD Carga un programa desde un grabador de cassette.

LOAD Carga un programa desde un periférico determinado.

MERGE Carga un programa desde un periférico determinado y lo une a otro programa ya almacenado en la memoria principal.

(2) VERIFICACION DEL PROGRAMA

Listar el programa por pantalla o mediante la impresora : ejecutar los siguientes comandos:

LIST n-m: Lista las líneas comprendidas entre "n" y "m" en la pantalla.

LLIST n-m: Lista las líneas comprendidas entre "n" y "m" en la impresora.

(3) CORRECCION DEL PROGRAMA

Editor de pantalla

El editor de pantalla permite posicionar el cursor en cualquier lugar de la pantalla y suprimir, insertar, corregir o renumerar líneas de programa.

1) Usando el comando **LIST**, listar la parte del programa al cual la línea a editar es la contenida en la pantalla.
2) Posicionar el cursor en el punto a corregir.
3) Realizar la correspondiente corrección.
4) Situando el cursor a la izquierda de la fila que acaba de ser modificada apretar la tecla [RETURN].

Nota 1: Si no se pulsa [RETURN], la modificación no se produce.
Nota 2: Se pueden copiar líneas de programa modificando su número de línea y pulsando [RETURN]. Sin embargo, la línea original queda sin cambios.

Función de editar	Operación de editar	Ejemplo
Modificación del contenido de una línea del programa. Ej. Corregir "PRIMT" en "PRINT"	(1) Posicionar el cursor en el carácter a corregir, usando las teclas de control de cursor ([→][←][↑][↓]). (2) Sobreimprimir el carácter corregido (3) Presionar la tecla [RETURN].	PRIMT■ "M" debe ser sustituido por N (■ indica el cursor). PRI■T Utilice [←] para colocar el cursor en "M". PRIN■ Sobreimprimir el carácter corregido (N). El cursor se situará en la posición siguiente.
Supresión de un carácter de una línea Ej. Corregir "PRIXNT" en "PRINT"	(1) Usando las teclas de control de cursor, colocar el cursor en el carácter a suprimir. (2) Presionar la tecla [DEL]. (3) Presionar la tecla [RETURN].	PRIXNT■ "X" debe ser suprimido. PRI■NT Posicionar el cursor en "X". PRI■T Presionar la tecla [DEL].
Inserción de un carácter en una línea listada. Ej. Corregir "PRNT" en "PRINT"	(1) Usando las teclas de control de cursor, posicionar el cursor donde sea necesario insertar el carácter olvidado. (2) Presionar la tecla [INS]. (3) Teclear el carácter a insertar. (4) Presionar otra vez [INS]. (5) Presionar [RETURN].	PRNT■ Falta "I". PR■T Utilizar [←] para llevar el cursor a "N" PRNT Apretar [INS] para seleccionar el modo INSERT. PRINT Teclear el carácter "I". PRI■T Presionar [INS] para limpiar el modo INSERT.

Función de editar	Operación de editar	Ejemplo
Corrección de una línea del programa completa.	(1) Teclear la línea a reemplazar con el mismo número de línea reemplazada. (2) Presionar la tecla [RETURN]. El contenido íntegro de la línea antigua habrá sido restituido por el nuevo.	Teclear la línea corregida presionar [RETURN]. Cuando se liste o corra el programa la nueva línea habrá reemplazado a la antigüa.
Supresión de una línea completa de programa.	(1) Teclee el número de línea a suprimir. (2) Presione [RETURN].	Teclear 10 presionar [RETURN]. La línea 10 quedará suprimida.
Supresión de más de una línea del programa.	(1) Ejecute el comando DELETE.	Teclear DELETE 20-60 [RETURN]. Esto suprimirá las líneas numeradas entre 20-60 cuando el programa sea listado o cuando corra.
Inserción de una línea entre otras dos (numeradas "n" y "m")	(1) Teclear la línea a insertar con un número de línea k (n < k < m) asignado a ella, entonces presione RETURN.	
Renumeración de líneas	(1) Ejecutar el comando RENUM.	Teclear RENUM 100, 10. Presionar [RETURN]. Esto renumerará las líneas con incrementos de 10 empezando en "100".

(4) Ejecución del programa

1) Para comenzar la ejecución de un programa ya entrado, teclear el comando RUN (apretar la tecla F5).
2) Para interrumpir la ejecución de un programa, presionar simultáneamente las teclas [CTRL] y [STOP].
   *Para reiniciar la ejecución detenida desde el punto de interrupción, entrar el comando CONT.
   *Sin embargo, no podrá reiniciar la ejecución del programa desde un punto de interrupción si ha realizado alguna corrección en el programa detenido.
3) Para detener temporalmente la ejecución de un programa, pulsar la tecla [STOP] (Mientras la ejecución del programa está en pausa, ninguna otra tecla diferente a [STOP] es operativa).

*Para reiniciar la ejecución del programa, utilizar la tecla [STOP] por segunda vez.

(5) Mensaje de error

Si durante la ejecución de un programa, se produce algún error, en la pantalla aparecerá un mensaje de error, de la siguiente forma:

Mensaje de error IN número de línea donde se ha producido.
(Ej.: **Syntax error in 10**)

1) Si se produce algún error, utilizar el Apéndice G para determinar la causa del mismo.
2) Una vez determinada la causa del error, corregirlo siguiendo las instrucciones descritas en Editor de pantalla.

(6) Grabación de un programa

Para grabar un programa en una cinta de cassette o en otro dispositivo periférico, utilizar los siguientes comandos:

CSAVE Guarda el programa en cinta de cassette.
SAVE Guarda el programa en un periférico determinado.

Ejemplo: Para este ejemplo, al archivo se le ha dado el nombre de "AGENDA" para poderse guardar. Para grabar el programa en cinta, APRETAR LA TECLA DE GRABACION Y PLAY DE SU GRABADOR y entrar:

CSAVE "AGENDA" y pulsar [RETURN]

El grabador comenzará automáticamente a grabar el programa. Cuando el grabador se detenga, el programa ya está grabado.

Para verificar si el programa está grabado...

Rebobinar la cinta y presionar PLAY del grabador, a continuación, escribir : CLOAD? "AGENDA" y pulsar [RETURN].
Cuando la computadora encuentra el programa en la cinta, escribirá en la pantalla: "Found: AGENDA"
Si el programa ha sido grabado incorrectamente aparecerá "Verify Error" en la pantalla. Después de la comprobación del grabador (ver Apéndice J) debera repetir la sentencia CSAVE.
Si no aparece error puede estar seguro de que el programa ha sido grabado correctamente.
Siempre es recomendable realizar el proceso de comprobación después de la utilización de la secuencia CSAVE.

NOTA: El nombre del archivo puede contener seis caracteres en el espacio comprendido entre las comillas (" ").

III - CONSTANTES Y VARIABLES

1. TIPOS DE DATOS

BASIC maneja los valores numéricos y alfanuméricos como datos. Los datos numéricos a su vez se clasifican en varios tipos. La siguiente tabla ilustra al respecto:

Valores numéricos
- Enteros
- Números reales
  - De simple precisión
  - De doble precisión

Datos alfanuméricos

* **Valores numéricos**
  Los valores numéricos utilizables en programas BASIC son enteros, números reales (positivos o negativos) y el cero.
  * Enteros
    Entre -32768 y +32767

  * Números reales
    Incluyen los números reales de simple y doble precisión

    Número real de simple precisión:
    Es un número real que puede tener seis dígitos significativos, en el rango entre -9.99999E+62 y +9.99999E+62. El rango válido del exponente es entre E+62 y E-64. El número más pequeño positivo que se puede representar es, entonces, 1.0 E-64.

    Números reales de doble precisión:
    Es un número real que puede tener 14 dígitos significativos, en el rango entre -9.9999999999999E+62 y +9.9999999999999E+62. El rango válido del exponente es entre E+62 y E-64. El número más pequeño positivo que se puede representar es, entonces, 1.0 E-64.

* **Datos alfanuméricos:**
  Son conjuntos de caracteres de hasta 255 elementos.

2. CONSTANTES

Las constantes son datos fijos o valores invariables o ítems utilizados en programas. Los diversos tipos de constantes son los siguientes:

- Constante numérica
  - Constante entera
    - Constante decimal
    - Constante hexadecimal
    - Constante octal
    - Constante binaria
  - Constante real
    - Simple precisión
      - De punto fijo
      - De punto flotante
    - Doble precisión
      - De punto fijo
      - De punto flotante
- Constante alfanumérica

(1) CONSTANTE NUMERICA

Una constante númerica negativa debe estar siempre precedida por un signo negativo (-). Una constante númerica positiva no requiere, en cambio, ir precedida por un signo positivo (+).

1) Constante entera

Las constantes enteras pueden expresarse en notación decimal, hexadecimal, octal y binaria.

Constante decimal:

* Es una valor decimal en el rango entre -32768 a +32767
* Es un número entero o real en el rango entre -32768 a +32767, con el sufijo "%". En los números reales, las fracciones decimales se truncan.
  (Ejemplo) 123 -567 12.7%

Constante hexadecimal:

* Un número hexadecimal se expresa con caracteres de 0 a 9 y de A hasta F, con el prefijo &H, y tienen un rango desde &H0 hasta &HFFFF. Las letras A hasta F mayúsculas corresponden a los números decimales desde 10 hasta 15.
* Hex &H0 hasta &H7FFF son 0 hasta 32767 en decimal; hex &H8000 hasta &HFFFF son -32768 hasta -1 en decimal.
  (Ejemplo) &HFF: 255 en decimal
  &HFFFE: -2 en decimal

Constante octal:

* Un número octal se expresa con caracteres entre 0 y 7, con el prefijo

&O, y tienen un rango entre &O0 y &O177777.

* Octal &O0 hasta &O077777 son 0 hasta 32767 en decimal; octal &O100000 hasta &O177777 son -32768 hasta -1 en decimal.
  (Ejemplo) &O377: 255 en decimal
  &O177776: -2 en decimal

Constante binaria:

* Un número binario se expresa con ceros y unos con el prefijo &B, y tiene un rango entre &B0 y &B1111111111111111 (16 unos).
* Binario &B0 hasta &B111111111111111 (15 unos) son 0 hasta 32767 en decimal; &B1000000000000000 hasta el número &B1111111111111111 (16 unos) son -32768 hasta -1 en decimal.
  (Ejemplo)
  &B11111111: 255 en decimal
  &B1111111111111110 es -2 en decimal

2) Constante real de simple precisión

Constante real de punto fijo

* Es un número real que tiene seis o menos dígitos significativos. Si tiene más de 6 dígitos significativos, es un número real de doble precisión.
* Es un número real o entero con el sufijo signo de admiración (!). Si tiene 7 o más dígitos significativos, el séptimo se redondea al valor más próximo.
  (Ejemplo) 9.87 0.0000345
  44.44986869! (el séptimo dígito significativo, 6, se redondea, resultando 44.4499.)

Constante real de punto flotante

* Consiste en una mantisa con seis o menos dígitos significativos y un exponente representado usando una E. El rango válido es desde -9.99999E+62 hasta +9.99999E+62, y el rango válido del exponente es desde +62 a -64. Un valor en punto flotante con más de seis digitos significativos en su mantisa es un número de doble precisión.
  (Ejemplo) 1.234E +23
  mantisa exponente

3) **Constantes reales de doble precisión**

Constante real de punto fijo

* Es un número real que tiene siete a catorce dígitos significativos. Si tiene 15 o más dígitos significativos, el décimo quinto se redondea al valor más próximo, y queda un número real de 14 dígitos significativos.

* Es un número real o entero con el sufijo numeral (#). Si tiene 15 o más dígitos significativos, el décimo quinto se redondea al valor más próximo, y queda un número real de 14 dígitos significativos.
  (Ejemplo) 9.873333345
  2344555555500
  44.449#

Constante real de punto flotante

* Consiste en una mantisa con siete a catorce dígitos significativos y un exponente representado usando una E. El rango válido es desde el número -9.9999999999999E+62 hasta el número +9.9999999999999E+62, y el rango válido del exponente es desde +62 a -64.
  (Ejemplo) 1.2345678E +23
  mantisa exponente

* Cuando el exponente se representa con una "D" en vez de una "E", la constante real de punto flotante es un número real de doble precisión, aún cuando su mantisa sea de 6 o menos dígitos significativos. El rango de validez es entre el número -9.9999999999999D+62 hasta +9.9999999999999D+62. El rango válido del exponente es desde +62 a -64.
  (Ejemplo) 1.23D +23
  mantisa exponente

(2) CONSTANTE ALFANUMERICA

Una constante alfanumérica consiste en un conjunto de caracteres con no más de 255 elementos encerrados entre comillas(" ").

* Un par de comillas (" ") con nada entre medio se llama alfanumérico nulo, y se lo trata como una constante alfanumérica.

* Un par de comillas (" ") con uno o más espacios entre medio es un alfanumérico de código ASCII 32 y se lo trata como una constante alfanumérica diferente al alfanumérico nulo (código ASCII 0).
* Un alfanumérico que contiene números entre comillas, como ser "123", se trata como constante alfanumérica (no como números). Luego, no se pueden realizar operaciones aritméticas con él.
  (Ejemplo) "Talent MSX"

3. VARIABLES

Una variable es una posición asignada de memoria usada para almacenar datos. Tiene su propio nombre de variable compuesto de un caracter o grupo de caracteres que se refiere al dato almacenado.

Puede asignar un valor a una variable ejecutando una asignación (LET), o las sentencias **INPUT** o **READ** (se reserva una posición de memoria para la variable, y el valor se transfiere a dicha posición).

Si se hace referencia a una variable antes de tener asignada un valor, tendrá un valor cero si es una variable numérica, y un alfanumérico nulo, cuando es una variable alfanumérica.

(1) TIPOS DE VARIABLES

Las variables se clasifican en los siguientes tipos, dependiendo del dato que almacenan. Cada tipo de dato debe almacenarse en su tipo respectivo de variable.
Una variable que se le puede asignar un solo valor por vez se la llama variable simple. Una variable que se le puede asignar más de un dato, se llama variable con subíndices o matrices.

- Variable simple
  - Variable numérica
    - Tipo entera
    - Tipo real
      - Simple precisión
      - Doble precisión
  - Variable alfanumérica (Caracteres)
- Variable con subíndices
  - Variable numérica
    - Tipo entera
    - Tipo real
      - Simple precisión
      - Doble precisión
  - Variable alfanumérica (Caracteres)

(2) NOMBRE DE VARIABLE

Un nombre de variable queda especificado con uno o dos caracteres alfanumérico y un símbolo de declaración de tipo de variable. A continuación damos algunas reglas de los nombres de variables.

1) El primer caracter debe ser una letra. El segundo caracter puede ser una letra o número.

Ejemplo: A3=2 Se le asigna el valor 2 a A3
3A=2 No se admite 3A como nombre de variable

2) Si se utilizan más de dos caracteres en un nombre de variable, solamente se toman en cuenta los dos primeros, con el tercero y los demás quedando ignorados.

Ejemplo: A3BC=2 Se le asigna el valor 2 a A3.

3) Todos los espacios y/o símbolos graficos se ignoran

Ejemplo: A 3=2 Se le asigna el valor 2 a A3

4) Las minúsculas utilizadas en los nombres de variables son convertidas en mayúsculas.

Ejemplo: a3=2 Se le asigna el valor 2 a A3

5) No se pueden utilizar como nombres de variables a las palabras reservadas por BASIC (como ser comandos, sentencias, funciones o nombres de operadores). Para ver la lista de palabras reservadas por BASIC, vea el Apéndice H.

Ejemplo: AUTO3=2 El nombre de variable "AUTO3" no se puede utilizar, ya que contiene la palabra reservada "AUTO".

6) Agrege un sufijo de declaración de tipo de variable para especificar el tipo de variable.

% Variable numérica entera
! Variable numérica simple precisión
Variable numérica doble precisión
$ Variable alfanumérica

* Las variables de igual nombre pero distinto sufijo de declaración de tipo de variable, se consideran variables diferentes.

Ejemplo: A%, A! y A son tres variables numéricas diferentes.

* Si no se utiliza el sufijo de declaración de tipo de variable, el sistema supone que es una variable de tipo real de doble precisión. Una variable cuyo tipo ha sido declarado por **DEFINT** (define entero), **DEFSNG** (define simple), **DEFDBL** (define doble), o **DEFSTR** (define string (alfanumérico)) se considera como que es del tipo de variable declarado por dichas sentencias.

Ejemplo: A# es el mismo nombre de variable que A. Si se especifica **DEFINT A** en un programa, A% es el mismo que A.

(3) VARIABLES CON SUBINDICE

Los conjuntos de datos se pueden manipular más fácilmente utilizando variables con subíndices o matrices. Llamamos matriz a un conjunto de variables con subíndice agrupadas bajo el mismo nombre, y los datos individuales se llaman elementos de la matriz.

Formato: Nombre (Subíndice) Matriz de una dimensión
Nombre (Subíndice,Subíndice) Matriz de dos dimensiones
Nombre (Subíndice,Subíndice,...)
(n elementos) Matriz n dimensional

1) La dimensión de una matriz se representa con un número de subíndices encerrados entre paréntesis (y separados por comas) siguiendo al nombre de la variable.

2) El rango de los subíndices va desde cero hasta la máxima cantidad permitida por la memoria disponible.

3) El tamaño (número de elementos) y dimensión de una matriz se especifica con la sentencia DIMensión.

Ejemplo: 10 DIM A(5,2)
Esta sentencia **DIM** declara una matriz bi-dimensional llamada A, en donde se pueden utilizar hasta (5+1) x (2+1)=18 elementos pueden utilizarse.

* Se pueden utilizar matrices de hasta tres dimensiones sin declararlas en una sentencia **DIM**. En este caso, se le asigna once elementos (de 0 a 10) para cada dimensión en forma automática.

(4) POSICIONES DE MEMORIA (CAPACIDAD) DISPONIBLE PARA CADA TIPO DE VARIABLE

Variable	Variable simple	Matrices
Tipo entero	5	5+2*(número de elementos) + 2*(dimensión)+1
Tipo real de simple precisión	7	5+4*(número de elementos) + 2*(dimensión)+1
Tipo real de simple precisión	11	5+8*(número de elementos) + 2*(dimensión)+1
Tipo alfanumérico	6 + (cantidad de caracteres)	5+3*(número de elementos) +2*(dimensión) + 1 + (número total de caracteres en los alfanuméricos contenidos en cada elemento)

(5) VARIABLES DEL SISTEMA

MSX BASIC tiene las siguientes variables de sistema reservadas para su propio uso:

TIME: Esta variable se refiere al valor inicial de un reloj de interrupciones que se incrementa en uno en intervalos de 1/50 de segundo. Este reloj puede inicializarse asignando el valor deseado a esta variable.

SPRITE$(n): Esta matriz alfanumérica se refiere a los diseños de los sprites.

VDP(n): Esta matriz numérica se refiere a los valores de los registros de la VDP.

(6) CONVERSION DE TIPOS DE VARIABLE

Un dato del tipo numérico puede ser convertido en otro dato del tipo numérico como se necesite (la conversión entre datos numéricos y alfanuméricos se maneja con las funciones **STR$** y VAL).

1) Si un valor numérico de cierto tipo se transfiere a una variable de otro tipo, se almacena el valor convertido del dato de acuerdo al tipo de variable asignado a la misma.

Ejemplo: A%=1.234 El entero 1 se asigna a A%

2) En las operaciones lógicas, los valores numéricos se convierten a tipo entero, y el resultado obtenido también es del tipo entero.

Ejemplo: A=NOT 1.234 El valor real 1.234 se convierte en el entero 1, y el resultado de la operación NOT sobre 1 se asigna a la variable A.

* Cuando un valor real se convierte entero, todos los decimales se truncan.
  Si el resultado del redondeo excede el rango válido de los enteros (-32768 a 32767) se genera un error.

* Cuando un número real de doble precisión es convertido a uno de simple precisión, el séptimo dígito significativo y los subsiguientes se redondean al número más próximo, resultando un número real de 6 dígitos.

IV - OPERACIONES

1. EXPRESIONES

Una expresión es una combinación de constantes, variables y/o funciones relacionadas por operadores. Una constante, variable o función sin ningún operador puede también denominarse expresión.

1) **Expresiones numéricas y alfanuméricas**
   El resultado de una operación especificado por una expresión puede ser un valor numérico o un conjunto de caracteres (valor alfanumérico). Las expresiones que producen resultados numéricos se llaman expresiones numéricas y las que producen conjuntos de caracteres, expresiones alfanuméricas.

2) **Tipos de expresiones numéricas**
   Las expresiones numéricas se subdividen a su vez en expresiones aritméticas, relacionales y lógicas. Todas estas expresiones producen resultados numéricos.

2. EXPRESIONES ARITMETICAS

Una expresión aritmética consiste en una o más constantes, variables, funciones numéricas conectadas por los operadores aritméticos. Siempre producen resultados numéricos.

Operador aritmético	Operación	Formato
+	Suma	X+Y
-	Resta	X-Y
*	Multiplicación	X*Y
/	División	X/Y
^	Potencia	X^Y
-	Signo negativo	-X
\	División entera	X\Y
MOD	Resto	X MOD Y

(1) **División entera**

* Para la división entera (\), todos los números reales son convertidos a entero, redondeando el primer dígito significativo, antes de efectuar la división.

* El cociente de una división entera es un entero, con todos los decimales truncados.
  Ejemplo: A=11.24\3 Se asigna el valor 3 a la variable A.

* El resto de una división entera es un entero y se obtiene utilizando el operador MOD.
  Ejemplo: A=11.24 MOD 3 Se asigna 2 a la variable A

(2) **División por cero**

Si se intenta dividir por cero, se genera un error. Se imprime el mensaje **"Division by zero"** en pantalla, y la computadora vuelve al modo Comando.

(3) **Potencias de cero**

La potencias de cero dan como resultado, en general, cero.
La potencia cero de cero (0^0) es uno (1).
Potencias negativas de cero generan error. Ver "División por cero".

(4) **Desborde**

Si el resultado de una asignación u operación aritmética excede el rango válido de la variable a la cual se le transfiere el resultado, se genera un desborde. El mensaje **"Overflow"** aparece en pantalla, y la computadora vuelve al modo Comando.

3. EXPRESIONES RELACIONALES

Una expresión relacional consiste de números o alfanuméricos conectados por operadores relacionales. Siempre produce un resultado numérico: -1 por verdadero, 0 por falso.
Las expresiones relacionales se usan principalmente para comparar el valor de dos datos en la sentencia IF.

Operador relacional	Significado	Formato
=	Igual	X=Y
<> o ><	Distinto	X<>Y
<	Menor que	X<Y
>	Mayor que	X>Y
<= o =<	Menor o igual	X<=Y
>= o =>	Mayor o igual	X>=Y

Ejemplos: IF X=Y THEN 100 ELSE 200
Si X es igual a Y, el control pasa a la línea número 100. De otro modo, pasará a la línea 200.
A=X=Y
El primer signo igual (=) es un símbolo de asignación, mientras que el segundo es un operador relacional. Si X es igual a Y, el valor -1 se asigna a la variable A. Si X, por el contrario, no es igual a Y, el valor asignado es 0.

(1) Comparaciones de datos alfanuméricos
En las expresiones relacionales, los datos alfanuméricos se comparan caracter por caracter, comenzando desde el primer caracter de cada alfanumérico. Dos valores alfanuméricos son iguales cuando cada caracter del primer alfanumérico es idéntico a su contraparte en el segundo alfanumérico. Si dos alfanuméricos no son iguales, el que contenga el caracter de mayor código ASCII se considera como el mayor. Si dos alfanuméricos tienen longitudes diferentes, el más largo se considera el mayor. Los espacios se cuentan cuando se computa la longitud del alfanumérico.

Ejemplos: "ABCDEF" es igual a "ABCDEF"
"AA" es menor que "AB"
"ABCDEF" es mayor que "ABCDE"
"AA " es mayor que "AA "
"A A" es menor que "AA "

(2) No es posible comparar datos numéricos con alfanuméricos.

La comparación siempre se efectúa entre datos numéricos y datos numéricos, o entre datos alfanuméricos y datos alfanuméricos.

4. EXPRESIONES LOGICAS

Una expresión lógica contiene una o más constantes, variables y/o funciones numéricas, unidas con operadores lógicos. Siempre producen resultados enteros.

Las expresiones lógicas se utilizan para comparar más de una expresión relacional, fundamentalmente en las sentencias IF, o para efectuar manipulaciones de bits u operaciones Booleanas.

Operador lógico	Significado	Formato
NOT	Negación (no)	NOT X
AND	Producto lógico (y)	X AND Y
OR	Suma lógica (o)	X OR Y
XOR	O exclusivo	X XOR Y
IMP	Implicación	X IMP Y
EQV	Equivalencia	X EQV Y

Ejemplo: 10 IF X>10 AND X<100 THEN 100

Si el valor de la variable X es mayor que 10 y menor que 100, el programa continúa en la línea 100.

10 IF X>10 OR Y<100 THEN 100

Si el valor de la variable X es mayor que 10 o el valor de la variable Y es menor que 100, el programa continúa en la línea 100.

(1) Tablas de verdad de los operadores lógicos

X	NOT X
1	0
0	1

X	Y	X AND Y	X OR Y	X XOR Y	X IMP Y	X EQV Y
1	1	1	1	0	1	1
1	0	0	1	1	0	0
0	1	0	1	1	1	0
0	0	0	0	0	1	1

(2) Operaciones lógicas

Cada operación lógica se ejecuta luego de que todos los valores numéricos entre -32768 y 32767 se convierten a su

complemento a dos. Si se intenta realizar una operación lógica fuera de este rango, se genera un error.
Las operaciones lógicas se ejecutan sobre cada bit de los operandos.

Ejemplo: A=7 OR 8
Los complemento a dos de los valores 7 y 8 son respectivamente &B111 y &B1000. El resultado de la operación lógica OR sobre estos valores es &B1111 (15 en decimal). Por lo tanto, se asigna el valor 15 a la variable A.

5. EXPRESIONES ALFANUMERICAS

Una expresión alfanumérica consiste en más de un alfanumérico unido a otro(s) alfanumérico(s) con el signo más (+). El resultado de una expresión alfanumérica es siempre un alfanumérico.

Ejemplo: A$="ABC":B$="DE":C$=A$+B$
Se asigna el alfanumérico "ABCDE" a la variable C$.

6. FUNCIONES

MSX BASIC provee un conjunto de funciones preprogramadas que simplifican muchas variedades de programas. Devuelven el resultado de operaciones funcionales especiales (Ejemplos: raíz cuadrada, valor absoluto, funciones trigonométricas, etc.) sobre datos que se le dan (argumentos). Las funciones son del tipo numérico y alfanumérico.

(1) Funciones numéricas y alfanuméricas
Las funciones se clasifican en funciones numéricas y alfanuméricas de acuerdo al valor que devuelvan cuando se ejecutan.

(2) Funciones predefinidas y definidas por el usuario
Además de las funciones predefinidas que vienen incluídas en el MSX BASIC, el usuario dispone la facilidad de poder programar sus propias funciones. Para la definición de estas funciones se utiliza la sentencia **DEF FN.**

(3) Números reales utilizados como argumentos
Si se utilizan argumentos reales de doble precisión y/o enteros, serán tratados como variables reales de doble precisión en la operación de las funciones.

(4) **Enteros utilizados como argumentos**
Si se utilizan reales cuando se requieren enteros como argumentos, se truncan los decimales que puedan éstos tener (sin redondeo).

7. ORDEN DE PRIORIDAD DE LAS OPERACIONES

1. Operaciones encerradas entre paréntesis
2. Funciones
3. Potencias (^)
4. Signo negativo (-)
5. Productos (*) y cocientes (/)
6. División entera (\)
7. Resto (MOD)
8. Sumas (+) y restas (-)
9. Operaciones relacionales (<, > ,=, etc.)
10. Operación lógica negación (NOT)
11. Operación lógica producto (AND)
12. Operación lógica suma (OR)
13. Operación lógica O exclusivo (XOR)
14. Operación lógica equivalencia (EQV)
15. Operación lógica implicación (IMP)

Las operaciones de igual prioridad se ejecutan de izquierda a derecha.

V - INTERRUPCIONES

El propósito de una interrupción es indicarle al CPU de la computadora que debe suspender todo lo que esté realizando, procesar el evento o datos que se le ingresan, y luego reasumir las operaciones suspendidas.

Si la señal de interrupción no se utiliza, el programa principal siempre deberá verificar si un evento o una lógica externa está requiriendo servicio. Con una señal de interrupción, por otra parte, sólo es necesario indicar la rutina de servicio de interrupción que se desea activar por su nombre y las sentencias de habilitación al principio del programa.

Esto elimina la necesidad de programar rutinas de verificación para estos eventos y acelera la ejecución del programa principal.

Causas Posibles de Interrupciones y sus sentencias de llamado:

Interrupción por error	ON ERROR GOTO
Interrupción por teclas de función	ON KEY GOSUB
Interrupción por tecla STOP	ON STOP GOSUB
Interrupción por colisión de sprites	ON SPRITE GOSUB
Interrupción por disparador joystick (Interrupción por barra espaciadora)	ON STRIG GOSUB
Interrupción por intervalo de tiempo	ON INTERVAL GOSUB

El orden de prioridades en las interrupciones es desde el principio al final de esta lista.

Ejemplo: En los siguientes programas, el programa se bifurca hacia la línea 1000 cuando se presiona la barra espaciadora:

Usando interrupciones:

```
10   ON STRIG GOSUB 1000
20   STRIG(0) ON
30   TIME=0
40   FOR I=1 TO 5000
50   PRINT T,TIME-T
60   T=TIME
70   NEXT
80   END
1000 PRINT "PULSO ESPACIO"
1010 RETURN
```

Sin usar interrupciones:

```
10   TIME=0
20   FOR I=1 TO 5000
30   PRINT T,TIME-T
40   T=TIME
50   K$=INKEY$
60   IF K$=" " THEN GOSUB 1000
70   NEXT
80   END
1000 PRINT "PULSO ESPACIO"
1010 RETURN
```

VI - LENGUAJE DE MAQUINA

(1) DESARROLLO DE UN PROGRAMA EN LENGUAJE DE MAQUINA

Su computadora personal Talent MSX DPC 200 tiene el chip Z80A como CPU (Unidad Central de Procesamiento, el "cerebro" de cualquier computadora). Deberá utilizar el lenguaje de máquina utilizado por el Z80 en sus programas en dicho código. Para más detalles sobre código de máquina, lea la literatura dedicada al Lenguaje de Máquina del Z80.
De ahora en más supondremos que usted sabe programar el Z80A.

1) Utilice la sentencia **CLEAR** para reservar un área de memoria donde residirá su programa en código de máquina.

2) Genere su programa en código de máquina utilizando las sentencias **POKE** y **PEEK.**

3) Para grabar sus programas en código de máquina, utilice la sentencia **BSAVE.** Para cargarlos en memoria, **BLOAD.**

4) Para ejecutar sus programas en código de máquina, utilice la sentencia **DEF USR** y la función **USR.**

Nota: Un pequeño error en la codificación de su programa en código de máquina puede hacer que su sistema se bloquee, y deberá entonces apagar temporalmente su computadora antes de continuar. Por lo tanto, recomendamos fuertemente grabar su programa en cassette antes de ejecutarlo.

(2) TRANSFERENCIA DE DATOS UTILIZANDO LA FUNCION USR

USR número (argumento)

1) Los datos se transfieren entre un programa BASIC y un programa en código de máquina a través de argumentos. Los programas en código de máquina utilizan **direcciones de memoria** para ingresar los argumentos.

2) El valor en la dirección **F663H** puede ser 2, 3, 4 u 8, dependiendo del tipo de argumento:

Valor de **F663H**	Tipo de argumento
2	Entero
3	Alfanumérico
4	Número real de simple precisión
8	Número real de doble precisión

3) Los parámetros de la función USR se pasan a y desde los programas en código de máquina de la siguiente manera:

* **Número entero**
  Un entero se representa en forma de dos bytes de dígitos binarios, y se almacenan en el orden

"menos significativo-más significativo", comenzando en la dirección F7F8H.

* **Número real de simple precisión**
  Un número real de simple precisión se representa con un byte de exponente y 3 bytes de mantisa, o sea, un total de 4 bytes, y se almacena en memoria en el orden "exponente - mantisa", comenzando en la dirección F7F6H hasta la dirección F7F9H.
  El bit más significativo (MSB) del exponente especifica el signo del número (0 si es positivo, 1 si es negativo), y los siete bits restantes representan el exponente, desde E +62 hasta E -64.
  La mantisa se representa con un binario codificado decimal (BCD) de seis digitos.

* **Número real de doble precisión**
  Un número real de doble precisión se representa con un byte de exponente y 7 bytes de mantisa, o sea, un total de 8 bytes, y se almacena en memoria en el orden "exponente - mantisa", comenzando en la dirección F7F6H hasta la dirección F7FDH.

* **Alfanuméricos**
  Para los alfanuméricos, la longitud y la dirección de memoria donde está localizado, indicada en el orden "menos significativo - más significativo", se almacenan en el orden en que están escritos, comenzando en la dirección de memoria indicada por el valor de las direcciones de memoria F7F8H y F7F9H.

MSX BASIC

MSX BASIC es un lenguaje de alto nivel muy poderoso y versátil diseñado para ofrecer compatibilidad a nivel software para los diferentes sistemas MSX.
Esto hace que una gran variedad de paquetes de programas escritos en MSX BASIC puedan ser utilizados en su TALENT MSX, y le permite intercambiar software con sus amigos que posean sistemas MSX de otras marcas.
MSX BASIC es una versión extendida del BASIC standard, versión 4.5, y del G.W. BASIC, ambos desarrollados por MICROSOFT CORPORATION

NOTACION UTILIZADA EN LAS INSTRUCCIONES.

Corchetes [ ]

Significa que es un ítem opcional

Ejemplo:
AUTO [línea inicial] [,incremento]
Es equivalente a:
AUTO
AUTO línea inicial
AUTO ,incremento
AUTO línea inicial, incremento

..... repetición

Significa que puede haber varias repeticiones en una línea.

Ejemplo:
Constante [,constante...]
Es equivalente a:
Constante
Constante, constante
Constante, constante, constante
Etc.

Expresión entera

Cuando se especifica **expresión entera,** una expresión numérica (incluyendo variables y constantes numéricas) pueden utilizarse. Si se utiliza una variable real de doble o simple precisión en la expresión, el valor de la misma se convierte a formato entero. Si sobrepasa 32767 genera error de desborde (overflow).

Ejemplo:
CHR$(expresión entera)
A$=CHR$(65.23) toma el valor "A" para A$
A$=CHR$(X) También son válidas las variables reales de doble precisión.

ABS

ABS(expresión numérica)

Devuelve el valor absoluto de **expresión numérica**, o sea sin su signo (+ ó -).
El resultado es siempre positivo con un número de doble precisión.

Ejemplo:

```
10 FOR I=-2 TO 2
20 PRINT "EL VALOR ABSOLUTO DE";I;"ES";ABS(I)
30 NEXT I
40 END
RUN
EL VALOR ABSOLUTO DE-2 ES 2
EL VALOR ABSOLUTO DE-1 ES 1
EL VALOR ABSOLUTO DE 0 ES 0
EL VALOR ABSOLUTO DE 1 ES 1
EL VALOR ABSOLUTO DE 2 ES 2
Ok
```

ASC

ASC(expresión alfanumérica)

Devuelve un valor numérico que es el código **ASCII** del primer caracter de **expresión alfanumérica.**

Si **expresión alfanumérica** es nula, devuelve el mensaje **"Illegal function call"** (llamado incorrecto a función).

La operación inversa se realiza con **CHR$(X).**

Los caracteres gráficos de códigos **Ø a 31** devuelven un valor **ASCII=1**, que es el código del caracter encabezamiento de gráficos . Ver más detalles en la instrucción **CHR$.**

Si no se especifica **número de línea e incremento** asume los valores 1Ø para cada uno.

Si **número de línea** es seguido por una coma, pero **incremento** no es especificado, queda asumido el último incremento indicado en un comando **AUTO.**

Si **AUTO** genera un número de línea que ya está utilizado, se imprimirá un asterisco (*) después del número de línea mencionado, para prevenir al usuario de que cualquier ingreso reemplazará a la línea existente. No obstante esto, si se tipea un [RETURN] inmediatamente después del asterisco, la línea será conservada y se generará el número de línea próximo.

Se termina la ejecución del comando **AUTO** pulsando [CTRL]+[C] ó [CTRL]+[STOP].El **BASIC** vuelve a modo comando. La última línea es ignorada.

Todos las funciones del editor de pantalla (las teclas de edición y el cursor) están disponibles también cuando se activa el comando **AUTO.**

Ejemplo:

```
AUTO 10,5
10 A=1
15 B=2
20 C=3 [CTRL]+[C]
LIST
10 A=1
15 B=2
Ok
AUTO 10
10* [RETURN]
15* [RETURN]
20 C=3
25 [CTRL]+[C]
10 A=1
15 B=2
20 C=3
Ok
```

BASE

BASE (expresión entera)

Devuelve la primer dirección de las tablas de la memoria del procesador de pantalla (VDP RAM). La siguiente es la descripción de **expresión entera**, que deberá estar en el rango de **Ø a 19**:

Ø - base de tabla de imagen de pantalla para modo texto 40x24
1 - sin aplicación
2 - base de tabla de generador de diseños para modo texto 40 x 24
3 - sin aplicación
4 - sin aplicación
5 - base de tabla de imagen de pantalla para modo texto 32x24
6 - base de tabla de color para modo texto 32x24
7 - base de tabla generadora de diseños para modo texto 32x24
8 - base de tabla de atributos de sprites para modo texto 32x24
9 - base de tabla de diseño de sprites para modo texto 32x24
1Ø - base de tabla de imagen de pantalla para modo de alta resolución
11 - base de tabla de color para modo de alta resolución
12 - base de tabla de generador de diseños para modo de alta resolución
13 - base de tabla para atributos de sprites para modo de alta resolución
14 - base de tabla para diseño de sprites para modo de alta resolución
15 - base de tabla de imagen de pantalla para modo de multi-color
16 - sin aplicación
17 - base de tabla para generador de diseños para modo multi-color
18 - base de tabla para atributos de sprites para modo multi-color
19 - base de tabla para diseño de sprites para modo multi-color

Ejemplo:

```
10 FOR I=5 TO 8
20 PRINT "BASE(";I;")";TAB(12);
30 PRINT RIGHT$("000"+HEX$(BASE(I)),4)
40 NEXT I
50 END
RUN
BASE( 5 )    1800
BASE( 6 )    2000
BASE( 7 )    0000
BASE( 8 )    1B00
Ok
```

BEEP

Para generar un sonido "bip" de aproximadamente ØØ4 segs. Equivale a utilizar la orden **PRINT CHR$(7).**

Ejemplo:

```
10 FOR I=0 TO 5
20 BEEP
30 NEXT I
40 FOR I=0 TO 100:NEXT I
50 FOR I=0 TO 5
60 PRINT CHR$(7)
70 NEXT I
RUN
(Se oyen 5 "bips", una pausa, y luego otros 5 "bips")
Ok
```

BIN$

BIN$(expresión numérica)

Devuelve una cadena de caracteres que representa el valor binario de **expresión numérica**

La **expresión numérica** está en el rango de **-32768** a **32767.** Si **expresión numérica** es negativa, se utiliza el complemento a 2. O sea, **BIN$ (-n)** es lo mismo que **BIN$ (65536-n).**

El resultado se convierte en un número donde se han suprimido los ceros menos significativos.

Ejemplo:

```
10 A=123:B=234
20 PRINT "A = ";BIN$(A)
30 PRINT "B = ";BIN$(B)
40 PRINT "A AND B = ";BIN$(A AND B)
50 PRINT "A OR  B = ";BIN$(A OR B)
60 PRINT "A XOR B = ";BIN$(A XOR B)
70 PRINT "A EQV B = ";BIN$(A EQV B)
80 PRINT "A IMP B = ";BIN$(A IMP B)
90 FOR A=-32768! TO -32765!
100 PRINT
110 PRINT A;": ";BIN$(A)
120 PRINT -A-1;": O";BIN$(-A-1)
130 NEXT A
140 END
RUN
A = 1111011
B = 11101010
A AND B = 1101010
A OR  B = 11111011
A XOR B = 10010001
A EQV B = 1111111101101110
A IMP B = 1111111111101110

-32768 : 1000000000000000
 32767 : 0111111111111111

-32767 : 1000000000000001
 32766 : 0111111111111110

-32766 : 1000000000000010
 32765 : 0111111111111101

-32765 : 1000000000000011
 32764 : 0111111111111100
Ok
```

BLOAD

BLOAD " dispositivo[: nombre de archivo] " [, R] [, offset]

Para cargar un programa objeto (programa en lenguaje de máquina) desde un determinado **dispositivo.**

Si se incluye la opción [R], después de cargar el programa, éste comienza su ejecución de manera automática, desde la dirección indicada cuando se almacenó el programa mediante el **BSAVE.**

El programa objeto quedará almacenado en la dirección de la memoria especificada con **BSAVE.** Si también se especificara **Offset,** todas las direcciones indicadas en **BSAVE** serán desplazadas (offset), por dicho valor.

Si se omite **nombre de archivo,** se cargará el primer programa objeto que encuentre en el dispositivo.

Una vez detectado el programa a leer, en forma automática se indicará en la pantalla: **"FOUND: nombre del archivo"** (Hallado:...), y comienza el proceso de carga del mismo.

Si **nombre del archivo** no coincide con el que guarda el dispositivo, se indicará **"Skip :nombre del archivo almacenado"**(buscar el que siga)

Si el programa **no** ha sido guardado con **BSAVE, BLOAD** lo ignorará completamente.

Si se nota que la carga de la memoria con **BLOAD** no finaliza, es probable que los datos o el programa haya sido cargado en el área de trabajo de BASIC o el bloque de control de archivo. Debe prestarse atención al valor de **Offset** cuando se efectúa la carga.

Ejemplo:

```
BLOAD "CAS: LPN", R, 5
FOUND: LPN
(En este caso se carga el programa en código de
máquina en la dirección original más 5, y comienza la
ejecución del mismo en forma automática)
Ok
```

BSAVE

BSAVE " dispositivo [:nombre de archivo] " , **dirección de comienzo** , dirección del final [, **dirección de ejecución**]

Para salvar una imagen de memoria en el **dispositivo** indicado.

dirección de comienzo y **dirección de final** son las direcciones de la parte inicial y final del área a ser salvada en el dispositivo.

dirección de ejecución se utiliza para dejar registrado en el archivo desde qué dirección debe comenzar la ejecución del programa cuando éste sea cargado mediante un **BLOAD**, pero si ésta es omitida, la dirección de comienzo será considerada como **dirección de ejecución**.

Ejemplo:

BSAVE "CAS : GAME1", &HEØØØ, &HEØFF
(En este caso cuando se carga el programa con BLOAD este dará comienzo desde "&HEØØØ" (hexadecimal EØØØ)).

BSAVE "CAS : GAME1", &HEØØØ, &HEØFF, &HEØ2Ø
(En este caso cuando se carga el programa con BLOAD este dará comienzo desde "&HEØ2Ø").

CALL

CALL **nombre sentencia extendida** [(**argumento,argumento,.....**)]

Para llamar a la subrutina **nombre sentencia extendida** que está almacenada en un cartridge de memoria **ROM** o un dispositivo periférico.

Cuando a dicha subrutina se deban enviar valores, éstos se indicarán mediante los **(argumentos,...)**, debiendo respetar el órden establecido al momento de definir los valores de entrada en la misma.

'_' es una abreviatura de 'CALL'.

Para más detalles de las subrutinas extendidas, lea la guía de referencia del periférico respectivo

Ejemplo:

```
CALL TALK("A","B","C")
_TALK("A","B","C")
```

(En este caso se llama a la rutina que permite hablar a la computadora, si está instalado el cartucho ROM correspondiente.)

CDBL

CDBL(expresión numérica)

Convierte la **expresión numérica** en un número de doble precisión. Pero la precisión de la respuesta será la misma que tenía antes de la conversión, o sea, la misma cantidad de dígitos significativos.

Ejemplo:

```
10 B!=2.333333#
20 A#=CDBL(B!/1.7)
30 PRINT A#
RUN
 1.3725470588235
Ok
```

CHR$

CHR$(expresión entera)

Devuelve un caracter alfanumérico cuyo código ASCII es **expresión entera.**

CHR$ es comunmente utilizada para enviar un caracter especial a la pantalla.

Es la inversa de la función **ASC.**

Si **expresión entera** es un código de control, el caracter correspondiente a ese código no se muestra pero es ejecutada la función de ese código.

Cuando **expresión entera** no esté comprendida en el rango de **Ø a 255**, se presentará un mensaje de error por llamado a función incorrecto.

Sin embargo los códigos **Ø a 31** no devuelven el caracter gráfico correspondiente. Para estos caracteres utilizar las comillas o envíe primero el caracter de encabezamiento de gráficos (**CHR$(1)**).

Ver el apéndice **B** donde se detalla la "Tabla de Códigos de Caracteres".

Ejemplo:

```
10 FOR J=33 TO 37
20 PRINT "CARACTER CUYO ASCII:";J;"ES ";CHR$(J)
30 NEXT
40 PRINT "EL CODIGO ASCII DE 'A' ES";ASC("A")
50 END
RUN
CARACTER CUYO ASCII: 33 ES !
CARACTER CUYO ASCII: 34 ES "
CARACTER CUYO ASCII: 35 ES #
CARACTER CUYO ASCII: 36 ES $
CARACTER CUYO ASCII: 37 ES %
EL CODIGO ASCII DE 'A' ES 65
Ok
```

CINT

CINT(expresión numérica)

Convierte la **expresión numérica** en un número entero, truncando la parte decimal (sin redondeo).

Si **expresión numérica** no está en el rango de **-32768 a 32767**, ocurrirá un error de **"desborde"** (overflow).

Ejemplo:

```
10 B#=100.999
20 PRINT CINT(B#),CINT(B#*2)
30 B%=B#
40 PRINT B%,B%*2
50 END
RUN
 100          201
 100          200
Ok
```

CIRCLE

CIRCLE (X,Y) ó STEP(X,Y), radio[,color][,ángulo inicial][,ángulo final][,excentricidad]

Para dibujar una elipse (o circunferencia) en modo alta resolución. **X** e **Y** son las coordenadas del centro tomando como referencia la esquina superior izquierda.

Para más detalles de **X** e **Y**, ver **especificador** de coordenadas en la instrucción **PUT SPRITE.**

Si se especifica **STEP(X,Y)** el centro quedará determinado en forma relativa al último punto impreso.

El **radio** es el eje mayor de la elipse. Se mide en puntos de pantalla (pixels)

El **color** es el código de color de los puntos de la figura. El valor supuesto (default) es el mismo que el color frente especificado en la sentencia **COLOR.**

Los parámetros **ángulo inicial** y **ángulo final** deberán estar indicados en radianes entre **Ø** y **2*PI.** Si **ángulo inicial** o **ángulo final** son negativos, la elipse se conectará al centro con una linea y se consideran los ángulos como positivos(sentido antihorario).

La **excentricidad** indica la relación entre los ejes mayor y menor de la elipse. El valor supuesto es 1. Si excentricidad es mayor que 1, la elipse aparece en forma vertical, caso contrario, o sea si **excentricidad** es menor que 1, el eje mayor es el horizontal.

Ejemplo:

```
10 CLS:SCREEN 2:A=10
20 FOR X=256 TO 0 STEP-A
30 CIRCLE(X,X/1.33),A:CIRCLE(256-X,X/1.33),A
40 A=A+2 :NEXT
50 FOR I=.1 TO 2 STEP .2
60 CIRCLE(128,165),20,,,,I
70 CIRCLE(128,165),20,4,,,I
80 NEXT
90 IF INKEY$="" THEN 50
RUN
```

(Aparece un gráfico interesante... Pulse cualquier tecla para finalizar.)

CLEAR

CLEAR [memoria reservada para variables alfanuméricas [,dirección más alta de memoria]]

Para poner todas las variables numéricas a cero, las alfanuméricas a nulas, cerrar todos los archivos que se encuentren abiertos y opcionalmente ajustar el fin de memoria.

El contenido de las sentencias que comienzan con **DEF (DEF FN, DEF USR, DEFINT, DEFSNG, DEFDBL, DEFDBL,** etc.) se cancelan.

memoria reservada para variables alfanuméricas es el espacio reservado para las variables alfanuméricas cuyo tamaño normal es de **200** bytes, y debe ser una expresión entera.

dirección más alta de memoria es la dirección de memoria más alta disponible para ser utilizada por el **BASIC-MSX** y debe ser especificada como una expresión entera, en bytes.

El área entre el límite especificado por **dirección más alta de memoria** y la dirección **&HF380** no puede ser accedida por el BASIC-MSX y los programas en código de máquina que se almacenen allí no serán destruídos.

El área inicial disponible es **&HF380**, que es igual a la dirección máxima disponible de memoria

Ejemplo:

```
10 A=10
20 B$="TEST"
30 PRINT A,B$
40 CLEAR
50 PRINT A,B$
60 END
RUN
 10              TEST
 0
Ok
```

CLOAD

CLOAD ["nombre archivo"]

Para cargar en memoria un programa BASIC almacenado en cassette bajo el nombre **"nombre archivo".**

CLOAD cierra todos los archivos que se encuentren abiertos y borra el programa que reside en la memoria.

Si se omite el **"nombre archivo"** será cargado el primer programa que contenga el cassette.

Nombre archivo debe constar de 6 letras como máximo. Si se especifican 7 o más letras, serán ignoradas a partir de la séptima.

Para todas las operaciones de lectura del cassette, la velocidad de transferencia en baudios, es automáticamente ajustada a la velocidad con que fue grabado ya que en **CSAVE** (grabación) se determina si se desea grabar **a 1200** o **a 2400** baudios.

Ejemplo:

```
CLOAD "MUESTRA"
FOUND: MUESTRA
Ok
```

(Se carga en memoria el programa almacenado bajo el nombre "MUESTRA" y elimina todo programa o archivo anterior.)

CLOAD?

CLOAD?["nombre del archivo"]

Para verificar un programa **BASIC** grabado en un cassette, con uno que está en la memoria.

Generalmente, **CLOAD?** es utilizado inmediatamente después del comando **CSAVE**, para confirmar que el programa que se hallaba en memoria, ha sido grabado correctamente en el cassette.

Para verificar el programa deberá rebobinar la cinta hasta donde se encuentra el programa en el cassette, digite CLOAD?, pulse el botón PLAY de su grabador y recién la tecla RETURN.

Ejemplo:

CLOAD? "MUESTRA"
FOUND: MUESTRA
Ok
(En este caso se verifica el programa denominado MUESTRA con el que se encuentra en memoria y se lo encontró correcto. Caso contrario emite el mensaje **"Verify error"** (error en verificación))

CLOSE

CLOSE [[#]número archivo[,[#]número archivo,....]]

Para cerrar el canal y liberar el buffer correspondiente al **número archivo** indicado. Una vez cerrado puede utilizarse el mismo **número archivo.**

Se pueden cerrar más de un archivo por vez, y si no se especifica ningún **número archivo,** se cierran todos los canales.

Si no se especifica **número archivo** se cerrarán todos los canales abiertos.

También cierra los archivos: **NEW - CLEAR - CLOAD - CSAVE - LOAD - SAVE** y cuando se edita una línea.

Mientras opera con archivos, mantenga siempre activado el dispositivo correspondiente (cassette, diskette, etc.)

Cuando se cierra un archivo que se abrió para salida de datos (OUTPUT), todos los datos almacenados en el buffer son enviados antes del cierre al archivo. Todos los archivos deben ser cerrados para poder volver a utilizarlos.

Ejemplo:

```
10 SCREEN 2
20 FOR J=30 TO 9 STEP -.5
30 I=J*7-30:K=14-J
40 LINE (I+50,I)-STEP(K*4,K*4),15,B
50 NEXT J
60 OPEN "GRP:" FOR OUTPUT AS #1
70 PRESET(30,140)
80 PRINT #1,"CUADRADOS"
90 CLOSE
100 GOTO 100
RUN
```

(Se dibuja un cuadrado. Pulse [CTRL] + [STOP] para finalizar)

CLS

Para borrar todos los caracteres y figuras gráficas de la pantalla, excepto las figuras sprite.

En los modos texto de pantalla, la sentencia **CLS** no borra los indicadores de función al pie de la misma, y ubica el cursor en la posición de origen (home) (Ø,Ø).

En los modos gráficos de pantalla, la ejecución de la sentencia **CLS** causa que el color de fondo de la pantalla cambie al especificado por la sentencia **COLOR.**

El último punto referenciado (que utilizan los **STEP** de las funciones gráficas) no cambia con la sentencia **CLS.**

Ejemplo:

```
10 SCREEN 0:WIDTH 37:A$=CHR$(248):B$=CHR$(193)+
CHR$(192):KEY OFF
20 CLS:SOUND 1,32:SOUND 8,255:SOUND 3,0
30 SOUND 5,5:SOUND 7,249:SOUND 9,15:SOUND 10,15
40 FOR I=0 TO 24
50 CLS:LOCATE I,0:PRINT B$:SOUND 0,I*10:SOUND
2,I*10
60 LOCATE I,I^2/24:PRINTA$
70 FOR J=1 TO 5:NEXT
```

```
80 NEXT I:LOCATE 24,22:PRINT CHR$(210)
90 SOUND 0,0:SOUND 1,5:SOUND 2,0
100 SOUND 3,13:SOUND 4,255:SOUND 5,15
110 SOUND 6,30:SOUND 7,0
120 SOUND 9,16:SOUND 10,16
130 FOR I=1 TO 30:NEXT
140 SOUND 12,56:SOUND 13,0
150 END
RUN
```

(Un avión lanza una bomba. Se usa **CLS** para el efecto de animación)

```
Ok
```

COLOR

COLOR [color de frente] [, color de fondo] [, color de bordes]

Para definir el color de frente (para los caracteres o gráficos), color de fondo, y/o colores de borde de la pantalla.

color de frente:

En un modo texto el color de los caracteres de la pantalla se define con el **color de frente.** Una vez que se ejecuta la instrucción **COLOR,** todos los caracteres que están en la pantalla cambian al color especificado en **color de frente**

En un modo de pantalla gráfico, el color de frente especifica el color de cada figura. Este color es el utilizado cuando se omite en las sentencias gráficas (**PSET, LINE, CIRCLE, DRAW, PAINT,** etc.)

color de fondo:

En un modo texto, la ejecución de la sentencia **COLOR** cambia el color de fondo de la pantalla.

En un modo de pantalla gráfico, el color de fondo sólo cambia cuando se ejecuta una sentencia **CLS** después de la sentencia **COLOR.** Caso contrario, la única sentencia que cambia el color de fondo es cuando se omite el color en la sentencia **PRESET.**

color de bordes:

Cambia el color de los bordes de la pantalla, o sea el área de la misma donde no se puede imprimir ningún caracter o gráfico.

El argumento, para cada una de las partes, puede estar entre **Ø y 15.** Los colores que tiene la pantalla al momento de encender la computadora son **15, 4, 4,** respectivamente.

El color que se corresponde con cada uno de estos valores es el que sigue a continuación:

Ø:	transparente
1:	negro
2:	verde mediano
3:	verde claro
4:	azul oscuro
5:	azul claro
6:	rojo oscuro
7:	cian
8:	rojo mediano
9:	rojo claro
1Ø:	amarillo oscuro
11:	amarillo claro
12:	verde oscuro
13:	magenta
14:	gris
15:	blanco

Se puede omitir hasta **2** códigos. P.ej.: **COLOR 15** cambia el color-frente de los caracteres. Para omitir un código de color de los primeros, colocar una coma .P.ej: **COLOR ,,7** cambia el color-borde.

color de bordes no tiene efecto en modo texto (**SCREEN Ø**.

Ejemplo:

```
10 CLS:SCREEN 1
20 FOR I=0 TO 15
30 FOR J=0 TO 15
40 PRINT "COLOR";I;" ";J
50 COLOR I,J
60 FOR K=0 TO 300
70 NEXT K
80 NEXT J
90 NEXT I
100 FOR I=0 TO 15
```

```
110 COLOR ,,I
120 FOR K=0 TO 300
130 NEXT K
140 NEXT I
150 COLOR 15,4,7
160 END
RUN
```

(Cambia de color los caracteres de frente y de fondo. Luego cambian los colores del borde de la pantalla. Finalmente vuelve al color normal (frente blanco, fondo azul) pero con el borde Cian).

```
Ok
```

CONT

Para continuar con la ejecución de un programa después de provocarse un BREAK (interrupción) mediante un STOP , un END, con las teclas [CTRL] + [STOP] o porque ocurrió un error.

En general, esta sentencia se utiliza en la depuración de programas. Para ello, se detiene la ejecución del mismo y se imprimen las variables involucradas con un comando PRINT en modo directo, antes de continuar con la ejecución del programa con la orden CONT.

Cuando se modifica cualquier línea del programa, automáticamente se borran todas las variables y la sentencia CONT no puede utilizarse (el programa no puede continuar).

Ejemplo:

```
10 CLS:PRINT "**** TEST DE INTERRUPCION ****"
20 STOP
30 PRINT "SIGUE EJECUCION DE PROGRAMA..."
40 GOTO 40
RUN
**** TEST DE INTERRUPCION ****
Break in 20
Ok
CONT
SIGUE EJECUCION DE PROGRAMA...
```

(Pulse [CTRL] + [STOP])

```
Break in 40
Ok
```

COS

COS(expresión numérica)

Devuelve el coseno de **expresión numérica**, que debe estar expresado en radianes.

La función **COS** se calcula siempre en doble precisión. Si se necesita otro tipo de precisión, el ajuste se efectúa después del cálculo.

Ejemplo:

```
10 CLS:P=3.1415926536#
20 FOR J=0 TO P STEP P/6
30 C!=COS(J)
40 PRINT "COS";INT(J/P*180);TAB(9);"=";C!
50 NEXT J
60 END
RUN
COS 0     = 1
COS 29    = .866025
COS 59    = .5
COS 90    =-5.15221E-12
COS 120   =-.5
COS 150   =-.866025
COS 180   =-1
Ok
```

CSAVE

CSAVE "nombre del archivo" [,baudios]

Para grabar un archivo de programa **BASIC** en cinta de cassette, en formato binario comprimido.

Los archivos **ASCII** insumen más espacio, pero algunos tipos de accesos requieren que los archivos se hallen expresados en formato **ASCII**. Por ejemplo, un programa que se intente fundir (**MERGE**) en otro debe ser salvado en formato **ASCII**.

Sin embargo, los programas grabados en formato binario comprimido ocupan menos cinta y son más rápidos de cargar.

Los programas salvados en **ASCII** pueden ser leídos como archivos de datos y de textos del **BASIC**, en cuyo caso debe utilizarse la instrucción **LOAD**. Para más información sobre grabación en formato **ASCII** ver la instrucción **SAVE**.

nombre del archivo no puede omitirse y su longitud máxima es de 6 caracteres. Las letras que excedan de 6 caracteres son ignoradas.

baudios permite seleccionar la velocidad de grabación.

Puede tomar los valores de **1** (**1200** baudios) ó **2** (**2400** baudios). Con opción **2** se logra grabar más rápidamente pero es crítica la calidad de la cinta y del grabador.

El valor de **baudios** asumido por el sistema es **1** (**1200** baudios) o la última opción **baudios** selecionada.

baudios también puede seleccionarse con la instrucción **SCREEN**.

NOTA: Se debe preparar la cinta antes de grabar (rebobinar, etc.), ya que la computadora envía los datos apenas se pulsa la tecla [RETURN].

Ejemplo:

> **CSAVE "MUESTRA"**
> (Se graba en el cassette bajo el nombre de "MUESTRA" el programa residente en memoria con opción **baudios=1** (asumido, ya que se supone que es la primera grabación))
> **Ok**

CSNG

CSNG(expresión numérica)

Convierte la **expresión numérica** a un número de simple precisión. Cuando **expresión numérica** tiene 7 ó más dígitos significativos se redondean al séptimo dígito.

Ejemplo:

```
10 J=COS(3.14/4)
20 A=CSNG(J)
30 PRINT J
40 PRINT A
50 END
RUN
 .70738826916719
 .707388
Ok
```

CSRLIN

Devuelve el **número de línea** (coordenada vertical) donde está posicionado el cursor, con un rango de **Ø a 23.** Ø corresponde a la primer fila.

La diferencia con el comando **POS** es que éste determina el **número de columna** (coordenada horizontal) donde está posicionado el cursor, con un rango de **Ø a 39.**

En cambio el comando **LOCATE** ubica el cursor en las coordenadas horizontal y vertical que se le especifíquen.

Ejemplo:

```
CLS
Ok
print csrlin '-- segunda línea
2
Ok
print csrlin '-- quinta línea
5
Ok
print csrlin '-- octava línea
8
Ok
```

DATA

DATA constante [,constante....]

Para almacenar las constantes numéricas y alfanuméricas dentro de un programa, a las que se accede por medio del comando **READ.**

Los comandos **DATA** no son ejecutables y pueden ser colocados en cualquier parte del programa.

Una declaración **DATA** puede contener tantas constantes como las que quepan en una línea (separadas por comas), y cualquier número de comandos **DATA** pueden ser utilizados en un mismo programa.

El comando **READ** permite leer las constantes almacenadas en los **DATA**, en forma secuencial (por número de línea), y los datos allí contenidos pueden considerarse como una lista contínua de ítems sin tener en cuenta cuantos ítems hay en una línea, o en donde están ubicados dentro del programa.

Las **constantes** pueden ser constantes numéricas de enteras, de simple y de doble precisión o constantes alfanuméricas.

Las constantes alfanuméricas, deberán hallarse entre comillas sólo si contienen comas, dos puntos o espacios significativos adelante o atrás, caso contrario las comillas son innecesarias.

El tipo de variable (numérica o alfanumérica) dada con el comando **READ** debe coincidir con la correspondiente constante en la declaración **DATA.**

Si las constantes numéricas son leídas como alfanuméricas, quedarán almacenadas como variables alfanuméricas.

Si las constantes alfanuméricas son leídas como numéricas, ocurre un error.

No es necesario que las constantes numéricas coincidan en su tipo (entera, real simple o doble precisión) ya que **READ** efectúa la conversión correspondiente.

Las constantes pueden ser leídas desde el principio o a partir de una línea indicada por medio de la declaración **RESTORE.**

Ejemplo

```
 10 FOR J=0 TO 3
 20 READ A$
 30 PRINT TAB(5);A$
 40 NEXT J
 50 READ A,B,C,D
 60 PRINT A,B,C,D
 70 DATA YO DESEO
 80 DATA TENER,UNA COMPUTADORA
 90 DATA TALENT MSX
 100 DATA 10,&H10,&O10,&B10
 110 END
RUN
      YO DESEO
      TENER
      UNA COMPUTADORA
      TALENT MSX
 10              16
 8               2
Ok
```

DEF FN

DEF FN nombre [(parámetro[, [parámetro...])] = definición de la función

Para definir y nombrar una función escrita por el usuario .

nombre puede utilizarse cualquiera de los nombres permitidos para designar una variable. Este nombre precedido por **FN** queda definido como el nombre de la función.

parámetro son los nombres de las variables comprendidas en la definición de la función que será reemplazada por los argumentos, cuando ésta sea llamada. Los ítems en esta lista se hallan separados por comas.

El nombre de una variable utilizado en la definición de una función, puede o no aparecer en la lista de **parámetros.** Si aparece, el valor

del **parámetro** debe ser provisto cuando la función es llamada, de lo contrario se utiliza el valor actual de la variable.

definición de la función es una expresión que realiza la operación de la función, limitada a una sola línea. Los nombres de variables que aparecen en esta expresión sirven solamente para definir la función y no afectan a las variables del programa que tengan el mismo nombre.

Si en el nombre de la función se especifican **parámetros**, el tipo de expresión (numérica o alfanumérica) deberá coincidir con estos **parámetros** antes de ser devuelta a la declaración de llamada, pero si los del argumento no coinciden con ellos, ocurrirá un error y se emitirá el mensaje de"**Type mismatch**" (No concuerda el tipo de variable).

El comando DEF FN deberá ser ejecutado antes de que la función que él define pueda ser llamada, caso contrario ocurrirá un error y se emitirá el mensaje "**Undefined user function**" (Función del usuario no definida).

DEF FN no está permitida en el modo directo.

Ejemplo:

```
10 DEF FNLN(X)=LOG(X)/LOG(10)
20 PRINT " X     LN(X)     LOG(X)"
30 FOR J=.4 TO 1.4 STEP .2
40 PRINT USING"#.#";J;
50 PRINT USING" ##.#####";LOG(J);FNLN(J)
60 NEXT J
70 END
RUN
 X     LN(X)     LOG(X)
0.4 -0.91629 -0.39794
0.6 -0.51083 -0.22185
0.8 -0.22314 -0.09691
1.0  0.00000  0.00000
1.2  0.18232  0.07918
1.4  0.33647  0.14613
Ok
```

DEFINT/DEFSNG/DEFDBL/DEFSTR

DEFINT/DEFSNG/DEFDBL/DEFSTR caracter alfabético-caracter alfabético [,caracter alfabético-caracter alfabético....] ó DEFINT/DEFSNG/DEFDBL/DEFSTR caracter alfabético [,caracter alfabético]

Para definir una variable como de tipo entero, simple precisión, doble precisión o alfanumérica respectivamente.

DEFINT/SNG/DBL/STR declaran que las variables cuyo nombre comienzan con los caracteres especificados por **caracter alfabético** o con un caracter dentro del rango señalado son del tipo entera, reales simple precisión, reales de doble precisión y alfanuméricas respectivamente.

Los nombres de variables que contengan símbolos de declaración de tipo de variable (%, !, # o $) son del tipo indicado por sus respectivos símbolos.

Una especificación de caracteres de comienzo siempre tiene precedencia sobre una declaración **DEFxxx** para definir qué tipo de variable es.

Con la instrucción **CLEAR** también se cancelan todas las "**DEFxxx**" que la preceden. De ser necesario seguir operando con ellas, deberán redefinirse.

Por ejemplo, **DEFINT A,D-F** definen como variables enteras a **AB, D, E1**, etc.

Ejemplo:

```
10 DEFINT I
20 DEFSNG J
30 DEFDBL K
40 DEFSTR L
50 I=1/3:PRINT I
60 J=1/3:PRINT J
70 K=1/3:PRINT K
80  L="ABC":PRINT L
90 CLEAR
100 GOTO 50
RUN
 0
```

```
 .333333
 .33333333333333
ABC
 .33333333333333
 .33333333333333
 .33333333333333
Type mismatch in 80
Ok
```

DEF USR

DEF USR [dígito]= dirección de comienzo

Para especificar la **dirección de comienzo** de una rutina USR (rutina en lenguaje de máquina).

dígito puede ser cualquier dígito de **Ø a 9** y corresponde al número de la rutina USR cuya dirección está siendo especificada, o sea, se pueden definir hasta 1Ø rutinas USR simultáneamente.

Si se omite **dígito**, queda asumido que nos estamos refiriendo a la rutina **Ø de USR.**

Cualquier número de líneas de DEF USR pueden aparecer en un programa para redefinir las direcciones de comienzo de las rutinas y por lo tanto permitir el acceso a tantas rutinas de lenguaje de máquina como sea necesario.

Ejemplo:

```
10 CLEAR 200,&HDFFF
20 DEFINT A-Z
30 AD=&HE000
40 DEFUSR=AD:DEFUSR1=&HC3:DEFUSR2=&H1113
50 FOR J=0 TO 3
60 READ A$
70 POKE AD+J,VAL("&H"+A$)
80 NEXT J
90 DATA 23,23,34,C9
100 A=USR1(0):INPUT "b=";B
110 A=USR(B)
120 PRINT "b+1=";A:A=USR2(0)
130 END
RUN
(Se borra la pantalla por la rutina USR1)
b=? 66
b+1= 67 (Suena un BEEP por la rutina USR2)
Ok
```

DELETE

DELETE [número de línea inicial] [-número de línea final]

Este comando permite borrar líneas de un programa.

número de línea inicial indica cual es la primer línea desde donde se desea borrar. Si sólo se especifica **número de línea inicial** , se borra dicha línea.

número de línea final indica hasta qué línea se desea borrar.
Si sólo se especifica **-número de línea final**, se borrarán las líneas que van desde la primera hasta la línea especificada.

Si se especifican ambos, se borran las líneas entre **número de línea inicial** y **número de línea final.**

Si se utiliza un punto (.) en lugar de los número de línea, queda especificada la última línea ejecutada por BASIC. Cuando ocurre un error y el programa se detiene, la última línea ejecutada es la del error.

Luego de utilizar el comando **LIST** o **LLIST**, la última línea ejecutada es la última listada.

Si se omiten **número de línea inicial** y **número de línea final**, se emitirá el mensaje de error: **"Illegal function call"** (llamado a función incorrecto).

El **BASIC** siempre devuelve el nivel de comando después de ejecutarse un **DELETE.**

Ejemplo:

```
10 A=1
20 B=2
30 C=A+B
40 D=A*B
50 E=A/B
60 F=A-B
70 PRINT A,B,C,D,E,F
80 END
```

```
DELETE 30-60
Ok
LIST
10 A=1
20 B=2
70 PRINT A,B,C,D,E,F
80 END
Ok
```

DIM

DIM nombre variable con subíndice (máximo subíndice[,.....]) [,nombre variable con subíndice (máximo subíndice[,.....])...]

Para especificar el **máximo subíndice** del **nombre variable con subíndice** indicada y disponer su almacenaje de acuerdo a dichos valores.

Si el **nombre variable con subíndice** se utiliza sin un comando **DIM**, asume que tendrá **1Ø** elementos como máximo en cada una de sus dimensiones (de **Ø** a **9**).

máximo subíndice es una expresión numérica válida que no incluya a la propia variable que se dimensiona.

No debe incluírse un mismo nombre de variable en dos **DIM**, salvo si se usa instrucción **ERASE** entre uno y otro, caso contrario se genera el error **"Redimensioned array"** (redimensiona variable con subíndice).

Si en el desarrollo del programa se utiliza un índice mayor que el máximo especificado, ocurrirá un mensaje de error: **"Subscript out of range"** (subíndice fuera de rango).

El valor mínimo para un índice es **Ø**. Puede tener hasta **255** subíndices, pero nunca se pueden especificar por falta de memoria. El límite práctico es alrededor de **1Ø**.

Todas las variables definidas por **DIM** tienen un valor inicial de cero para variables numéricas y nulo ("") para las alfanuméricas.

Las variables con subíndice pueden eliminarse con las sentencias **ERASE** o **CLEAR**.

Cuando queda poca memoria, los arreglos innecesarios pueden borrarse con estas sentencias. Si necesita reinicializar algún arreglo (variable con subíndice) elíminelo temporariamente y luego redeffínalo con la sentencia **DIM.**

Ejemplo:

```
10 DIM A$(5,3)
20 FOR H=0 TO 4
30 FOR J=0 TO 2
40 READ A$(H,J)
50 PRINT A$(H,J);
60 NEXT J
70 NEXT H
80 READ A$(6,3)
90 DATA 1,3,5,2,q,w,e,r,t,y
100 DATA !,@,#,$,%,&,',(,),-
110 END
RUN
1352qwerty!@#$%
Subscript out of range in 80
Ok
```

DRAW

DRAW expresión alfanumérica

Para dibujar una figura de acuerdo al macro-lenguaje gráfico.

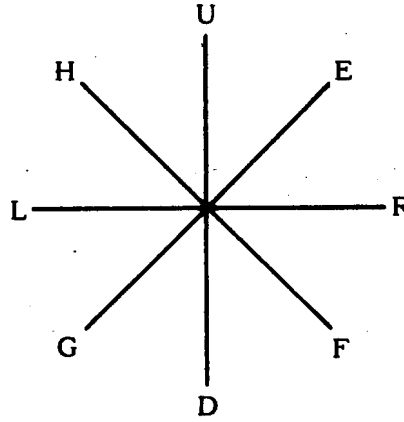
Los comandos del macro-lenguaje gráfico están contenidos en la **expresión alfanumérica.**

La **expresión alfanumérica** define un objeto que es dibujado cuando el BASIC ejecuta el comando **DRAW.**

Durante la ejecución el BASIC examina el valor de la **expresión alfanumérica** e interpreta el significado de cada letra comando del contenido de la misma.

Los siguientes comandos comienzan el movimiento desde el último punto referenciado. Este queda definido como la posición en que se imprimió el último punto en la pantalla, a través de las sentencias: **PSET, PRESET, CIRCLE, LINE** ó **DRAW.**

Un: Mueve hacia arriba
Dn: Mueve hacia abajo
Ln: Mueve a la izquierda
Rn: Mueve a la derecha
En: Mueve en diagonal arriba y a la derecha
Fn: Mueve en diagonal abajo y a la derecha
Gn: Mueve en diagonal abajo y a la izquierda
Hn: Mueve en diagonal arriba y a la izquierda



n en cada uno de los comandos precedentes indica la cantidad de unidades a imprimir y el número de puntos dibujados es n veces el factor de la escala (fijado por el comando S). Se acostumbra utilizar la instrucción PSET para el punto inicial de referencia.

Mx,y: Mueve en forma absoluta o relativa.

Si x o y tienen un signo positivo (+) o signo menos (-) antes de ellos, el movimiento será relativo. De lo contrario será absoluto o sea, toma como referencia el punto (0,0) de la pantalla.

Siendo la relación ancho-alto de pantalla igual a 1, es que 8 puntos horizontales son iguales en largo a 8 puntos verticales.

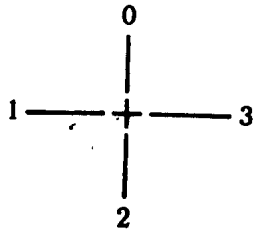
Los dos comandos prefijos siguientes pueden preceder a cualquiera de los movimientos enumerados:

Bn: Mueve pero no dibuja ningún punto.
Nn: Mueve y retorna a la posición original al finalizar.

También se hallan disponibles los comandos siguientes:

An: Gira las figuras dibujadas con U, D, L, R, E, F, G, H o M (sólo en formato relativo) en incrementos de 9Ø grados..

n puede variar de Ø a 3, donde Ø es Ø grados, 1 es 9Ø grados, 2 es 18Ø grados y 3 es 27Ø grados.



Cn: Determina el color n, variando de Ø a 15.
Sn: Determina el factor de la escala, variando de 1 a 255.

n dividido por 4 es el factor de la escala, es decir que si n=1, entonces la escala es de 1/4.

El factor de la escala multiplicado por la distancia dada por los comandos gráficos U, D, L, R, E, F, G, H y los comandos relativos M determinan la distancia movida. El valor normal es U que significa "relación:punto pantalla=punto comando(n)".

X variable alfanumérica; : Cumple una secuencia dentro de otra.

Ejemplo:

```
A$= "U8ØR8ØD8ØL8Ø":DRAW "XA$;"
```

En todos estos comandos, los argumentos n, x, o y pueden ser constantes numéricas o variables numéricas.

Siempre se requerirá el ";" cuando se utiliza una variable de esta manera, o en el comando X, pero sí es opcional entre comandos.

Los espacios son ignorados en las **expresiones alfanuméricas**; significa que se podrían emplear variables con un comando de movimiento de la siguiente manera:

```
X1=4Ø : X2=5Ø : DRAW "M+ =X1;,-=X2"
```

El comando X puede ser muy útil, ya que permite definir una parte de un objeto separada del total o dibujar una cadena de comandos de un largo mayor que 255 caracteres.

Ejemplo:

```
10 SCREEN 2
20 PSET(220,191),10
30 DRAW"U190"
40 FOR J=189 TO 1 STEP -4
50 A$="L"+STR$(J)+"D"+STR$(J-1)+"R"+STR$
(J-2)+"U"+STR$(J-3)
60 DRAW "XA$;"
70 NEXT J
80 GOTO 80
RUN
```

(Se dibuja un gráfico que demuestra la capacidad gráfica de su computadora... Pulse [CTRL] + [STOP] para finalizar)

END

Para terminar la ejecución de un programa, cerrar todos los archivos y devolver el control a nivel de comando, pudiendo colocarse en cualquier parte del programa.

La diferencia entre el comando STOP y el comando END, está dada porque este último no provoca la emisión del mensaje BREAK en la pantalla.

Si un programa finaliza en la última linea, el comando END es opcional. Sin embargo, en este caso no se cierran los archivos abiertos.

Ejemplo:

```
10 PRINT "*** EL PROGRAMA HA TERMINADO ***"
20 END
30 PRINT "NADA MAS"
RUN
*** EL PROGRAMA HA TERMINADO ***
Ok
```

EOF

EOF (número de archivo)

La función **EOF** permite consultar si se ha detectado el fin del archivo secuencial **número de archivo**, devolviendo -1 (verdad) si éste ha sido alcanzado, caso contrario devuelve un **Ø** (falso).

La función **EOF** se utiliza normalmente en declaraciones **IF...THEN**, para así dirigir el flujo del programa a la rutina correspondiente.

El archivo especificado por **número de archivo** debe ser abierto con la sentencia **OPEN** en el modo **INPUT** únicamente.

Es importante utilizar el **EOF** mientras se están efectuando lecturas a un archivo, a fin de evitar el mensaje de error: **"Input past end"** (lectura posterior al fin de archivo).

Ejemplo:

```
10 OPEN"CAS:DATA"FOR INPUT AS #1
20 IF EOF(1) THEN END
30 INPUT #1,N,R
40 PRINT "SQR(";N;")=";R
50 GOTO 20
RUN
```

(Lee el archivo llamado **DATA** en cassette e ingresa un par de valores numéricos por cada lectura hasta encontrar el fin de archivo. La orden **END** cierra el archivo.)

```
Ok
```

ERASE

ERASE nombre de variable [,nombre de variable....]

Para borrar las variables matriciales indicadas en **nombre de variable**, como así también su reserva de espacio en memoria.

Las matrices pueden ser redimensionadas con la sentencia **DIM** luego de haberse aplicado el **ERASE**, y el espacio previamente reservado en la memoria, puede ser nuevamente utilizado.

Si se trata de redimensionar una matriz sin haberlas borrado previamente por medio del comando **ERASE**, se producirá un error y

se emitirá el siguiente mensaje: **"Redimensioned array"** (matriz ya dimensionada).

Ejemplo:

```
10 DIM A(5)
20 FOR I=0 TO 5
30 READ A(I)
40 PRINT A(I);
50 NEXT:PRINT
60 ERASE A
70 FOR I=0 TO 9
80 PRINT A(I);
90 NEXT
100 END
110 DATA 1,2,3,4,5,6
RUN
 1  2  3  4  5  6
 0  0  0  0  0  0  0  0  0  0
Ok
```

ERL/ERR

Cuando se ingresa una subrutina para manejar errores, la variable **ERR** contiene el código para el error y la variable **ERL** contiene el número de línea en que éste fue detectado.

Las variables **ERR** y **ERL** son utilizadas usualmente en las declaraciones **IF...THEN** u **ON ERROR GOTO** para dirigir el flujo del programa dentro de la rutina para captar errores .

Si la declaración que provocó el error estaba en modo directo, **ERL** contendrá **65535** y para verificarlo utilice :

IF ERL = **65535** THEN.....

De lo contrario, utilice:

IF ERL = **número de línea** THEN....
IF ERR = **código de error** THEN....

Al ser **ERL** y **ERR** variables reservadas, ninguna de ellas puede aparecer a la izquierda del signo igual en una declaración **LET** (asignación). En los comparadores **IF** no puede figurar a la derecha del signo de comparación.

Cuando se utilize la sentencia **RENUM**, los números que figuran en estos comparadores son automáticamente reemplazados.

Vea el apéndice G para los código de error y sus mensajes.

Ejemplo:

```
10 CLS:PRINT"CALCULO DE 1/N Y 10^N":ON ERROR GOTO 80
20 INPUT "INGRESE N (fin=-1):";N
30 IF N=-1 THEN ON ERROR GOTO 0:END
40 A=1/N:B=10^N
50 PRINT "1/";N;"=";A
60 PRINT "10^";N;"=";B
70 GOTO 20
80 IF ERR=6 THEN PRINT "10^";N;"es un numero
excesivamente grande para mi!":RESUME 20
90 IF ERR=11 THEN PRINT "1/";N;"es DIVISION POR
CERO! Donde lo  aprendiste???":RESUME 20
100 PRINT"Que hiciste?":RESUME NEXT:GOTO 20
RUN
CALCULO DE 1/N Y 10^N
INGRESE N (fin=-1):? 2
1/ 2 = .5
10^ 2 = 100
INGRESE N (fin=-1):? 0
1/ 0 es DIVISION POR CERO! Donde lo  aprendiste???
INGRESE N (fin=-1):? 999
10^ 999 es un numero excesivamente  grande para mi!
INGRESE N (fin=-1):?-1
Ok
```

ERROR

ERROR código de error

Para simular que ocurre un error o permitir que los códigos de error sean definidos por el usuario.

El valor de **código de error** debe estar en el rango entre **Ø** y **255.**

Si el valor de **código de error** es igual a un código utilizado por el **BASIC**, la declaración **ERROR** simulará que ocurre dicho error y emitirá el mensaje correspondiente.

Para definir un código de error propio es recomendable utilizar un valor mayor a cualquiera de los utilizados por el BASIC para código de errores.

Es conveniente consultar el apéndice G para verificar la lista de códigos de error y sus correspondientes mensajes. (Es preferible utilizar los valores más altos disponibles, a efectos de que sea mantenida la compatibilidad cuando al BASIC se le agreguen más códigos de error).

Tanto los códigos de error definidos por el usuario como los definidos por el BASIC, pueden ser manejados convenientemente en una rutina para detectar errores.

Ejemplo:

```
10 ON ERROR GOTO 1000
:
120 IF A$="Y" THEN ERROR 250
:
1000 IF ERR=250 THEN PRINT"SEGURO?"
```

Si una declaración **ERROR** especificó un código para el que no ha sido definido su mensaje de error o para el cual no hay rutina de detección del error, se emitirá un mensaje por pantalla: **"Unprintable error"** (error no identificable) y se producirá una interrupción del proceso.

Ejemplo:

```
10 INPUT "ERROR NO.=";A
20 ERROR A
30 END
RUN
ERROR NO.=? 2
Syntax error in 20
Ok
RUN
ERROR NO.=? 10
Redimensioned array in 20
Ok
```

EXP

EXP(expresión numérica)

Devuelve **'e'** elevado a la potencia de **expresión numérica,** siendo **'e'** la base del logaritmo natural.(e=**2.718281...**)

expresión numérica deberá ser menor o igual que **145.Ø6286Ø85862.** Caso contrario, se imprimirá el mensaje de error: **'Overflow'** (Desborde).

Ejemplo:

```
10 PRINT " X     EXP(X) LOG(EXP(X))"
20 FOR J=.4 TO 1.4 STEP .2
30 PRINT USING "#.# ###.###";J;EXP(J);
40 PRINT LOG(EXP(J))
50 NEXT J
60 END
RUN
 X     EXP(X) LOG(EXP(X))
0.4    1.492 .39999999999985
0.6    1.822 .59999999999982
0.8    2.226 .79999999999987
1.0    2.718 .99999999999986
1.2    3.320 1.1999999999999
1.4    4.055 1.4
Ok
```

FIX

FIX(expresión numérica)

Devuelve la parte entera de la **expresión numérica** (fracción truncada).

FIX(X) es equivalente a la expresión: **SIGN(X)*INT(ABS(X)).**

La diferencia más grande entre **FIX** e **INT** es que **FIX** devuelve el próximo número superior para **X** negativa.

Ejemplo:

```
10 PRINT "  X        FIX(X)  INT(X)"
20 FOR J=-1.2345 TO 1.2345 STEP .5
30 PRINT USING "##.####";J;
40 B=FIX(J)
50 PRINT USING "######";B;
60 B=INT(J)
70 PRINT USING "######";B
80 NEXT J
90 END
RUN
  X        FIX(X)  INT(X)
-1.2345     -1     -2
-0.7345      0     -1
-0.2345      0     -1
 0.2655      0      0
 0.7655      0      0
Ok
```

FOR

FOR variable contador = valor comienzo TO límite [STEP incremento]

La instrucción **FOR TO !STEP!** reitera la ejecución de las instrucciones que están ubicadas entre **FOR TO !STEP!** y **NEXT** hasta que la **variable contador** cae fuera del rango del **límite.** Este conjunto de instrucciones comprendidas por las sentencias **FOR TO !STEP!** y **NEXT** lo denominamos **ciclo FOR-NEXT.**

variable contador es utilizada como contador. **valor comienzo** es el valor inicial del contador. **límite** es el valor final al que tiene que llegar **variable contador.**

Las líneas del programa ubicadas después del **FOR** son ejecutadas hasta encontrar su correspondiente instrucción **NEXT.**

Cuando esto sucede, se le suma el **incremento** (con su signo) a la **variable contador** y se verifica si llegó al valor **límite.** Si es así, ejecuta la siguiente instrucción después del **NEXT**; caso contrario, la ejecución del programa salta a la línea siguiente de la instrucción **FOR.**

variable contador deberá ser una variable numérica **sin** subíndice.

valor comienzo, límite e **incremento** son expresiones numéricas válidas.

Ejemplo:

```
40 FOR J = 1 TO 10
   ..........
90 NEXT J
```

La línea 40 especifica que la **variable contador** de este ciclo **FOR-NEXT** es **J.**

Inicialmente se le asignará a esta variable el valor 1, y será incrementada del hasta 10 durante el curso del proceso de repetición.

La línea 90 marca el final del ciclo **FOR-NEXT** con la palabra **NEXT**, y se refiere nuevamente a la **variable contador J.**

Ya que **J** tomará **10** diferentes valores durante el proceso de repetición, las instrucciones ubicadas entre las líneas 40 a 90 serán ejecutadas 10 veces, una por cada valor de **J.**

Cualquier instrucción **BASIC** puede aparecer como líneas dentro del ciclo **FOR-NEXT** y frecuentemente se hace uso dentro del mismo de la **variable contador.**

Ejemplo:

```
40 FOR J = 1 TO 10
50 PRINT J
   ..........
90 NEXT J
```

La línea 50 desplegará sobre la pantalla, cada valor de la **variable contador** a medida en que se va produciendo la repetición.

Está permitido que más ciclos **FOR** aparezcan dentro de un ciclo **FOR-NEXT.** Estos son llamados **ciclos FOR-NEXT anidados:**

Ejemplo:

```
40 FOR J = 1 TO 10
50 FOR K = 1 TO 5
   ..........
80 NEXT K
90 NEXT J
```

En este caso, el ciclo interno se repetirá 5 veces cada vez que se ejecuta el ciclo externo, o sea que las líneas 5Ø y 8Ø serán ejecutadas en un total de 5Ø veces.

Hay dos reglas esenciales para recordar cuando se construyan ciclos anidados FOR-NEXT:

1.-El ciclo interno debe tener su propia variable contador, diferente a la del ciclo externo (en el ejemplo anterior la variable contador del ciclo interior es K).

2.-El ciclo interior debe estar completamente contenido dentro del exterior, es decir que las líneas **FOR-NEXT** del ciclo interior deben estar ubicadas dentro las líneas **FOR-NEXT** del ciclo exterior.

Ejemplo QUE NO ESTA PERMITIDO:

```
1Ø FOR J = 1 TO 1Ø
2Ø FOR K = 1 TO 5
   ..........                    NO!
5Ø NEXT J
6Ø NEXT K
```

Los rangos de valores para la **variable contador** de un ciclo **FOR** pueden expresarse como variables numéricas o como expresiones numéricas.

Ejemplo:

```
4Ø FOR J = A TO (B + C)
```

Las variables A, B, y C deben ser definidas e inicializadas antes de que el programa llegue a la línea 4Ø, y de ésta resultará que J tomará valores desde A hasta B + C durante el proceso de repetición, con incrementos de 1.

El incremento de cada repetición del ciclo puede especificarse con valores diferentes a 1 por medio del operando opcional **STEP.**

Ejemplo:

```
4Ø FOR J = 1 TO 1Ø STEP 2
```

Cuando STEP no se especifica en un **ciclo FOR-NEXT,** el incremento normal será de 1.

Es posible realizar un ciclo cuyos valores se **decrementen,** para ello se deberá utilizar un **STEP negativo.** Si se especifica **STEP Ø,** el ciclo **FOR-NEXT** se convierte en un ciclo infinito (no se detiene a menos que se pulse [CTRL] + [STOP]

Ejemplo:

```
10 FOR J=10 TO 0 STEP -1
20 PRINT J;
30 NEXT J
40 END
RUN
 10  9  8  7  6  5  4  3  2  1  0
Ok
```

Si la variable que se le asigna al **NEXT,** no es la **variable contador** definida en el comienzo del **ciclo FOR-NEXT,** se cancelará la ejecución del programa con el mensaje de error: **"NEXT without FOR"** (NEXT sin su FOR).

Ejemplo:

```
10 K=5
20 FOR J=1 TO 10
30 PRINT J*K
40 NEXT K
RUN
 5
NEXT without FOR in 40
Ok
```

En caso de estar **mal especificado el rango** de la **variable contador,** de forma tal que desde el comienzo **valor inicial** es menor que **límite** e **incremento** mayor que cero, se ignora el ciclo.

Ejemplo:

```
10 FOR J=10 TO 0
20 PRINT J;
30 NEXT J
RUN
 10
Ok
```

Si dentro del ciclo **FOR-NEXT** se ejecuta una sentencia **CLEAR** o **MAXFILES,** el ciclo se da por finalizado.

EN RESUMEN....

Ejemplo.

Este programa genera una tabla de multiplicar sobre la pantalla.

El par de ciclos anidados entre las líneas **3Ø** a **8Ø** crean la tabla y el ciclo interior entre las líneas **4Ø** a **6Ø** calcula e imprime una fila de valores, siendo el ciclo exterior quien lleva al proceso a través de las 9 filas de la tabla.

La línea **5Ø** entonces, es ejecutada 81 veces, una por cada una de las 81 entradas de la tabla.

Ejemplo:

```
10 PRINT TAB(10);"TABLA DE MULTIPLICAR"
20 PRINT
30 FOR J=1 TO 9
40 FOR K=1 TO 9
50 PRINT USING "### ";J*K;
60 NEXT K
70 PRINT:PRINT
80 NEXT J
90 END
RUN
          TABLA DE MULTIPLICAR

  1   2   3   4   5   6   7   8   9

  2   4   6   8  10  12  14  16  18

  3   6   9  12  15  18  21  24  27

  4   8  12  16  20  24  28  32  36

  5  10  15  20  25  30  35  40  45

  6  12  18  24  30  36  42  48  54

  7  14  21  28  35  42  49  56  63

  8  16  24  32  40  48  56  64  72

  9  18  27  36  45  54  63  72  81

Ok
```

FRE

FRE(expresión numérica)
FRE(expresión alfanumérica)

FRE devuelve el número de bytes que hay en memoria sin ser utilizados por el **BASIC** y sus argumentos son ficticios, pudiendo tomar cualquier valor.

FRE(expresión numérica): devuelve el número de bytes de memoria disponibles, que pueden utilizarse para desarrollar programas en **BASIC**, almacenar archivos de texto, variables, etc.

FRE(expresión alfanumérica): devuelve el número de bytes de memoria disponible para almacenar variables alfanuméricas. Este área es la que forma el primer parámetro de la sentencia **CLEAR**. Para más detalles ver el Apéndice D.(mapa de memoria)

Ejemplo:

```
10 PRINT "BYTES LIBRES";FRE(0);
20 PRINT "AL COMIENZO"
30 FOR J=200 TO 1000 STEP 200
40 DIM A(J)
50 PRINT "BYTES LIBRES";FRE(0);
60 PRINT "PARA DIM A(";J;")"
70 ERASE A
80 NEXT J
90 END
RUN
BYTES LIBRES 23656 AL COMIENZO
BYTES LIBRES 22004 PARA DIM A( 200 )
BYTES LIBRES 20404 PARA DIM A( 400 )
BYTES LIBRES 18804 PARA DIM A( 600 )
BYTES LIBRES 17204 PARA DIM A( 800 )
BYTES LIBRES 15604 PARA DIM A( 1000 )
Ok
```

GOSUB

GOSUB número de línea

Para efectuar una bifurcación incondicional al comienzo de una subrutina, fuera de la secuencia normal del programa, ubicada en **número de línea**, dejando fijada, en forma automática, la dirección de retorno, como la siguiente a la de partida.

Una subrutina es un conjunto de instrucciones independientes que siempre terminan en una sentencia **RETURN.**

número de línea es la primera línea de una subrutina, la cual puede ser llamada cualquier cantidad de veces dentro de un programa.

Así mismo una subrutina puede ser ser llamada desde dentro de otra subrutina y este anidado de subrutinas está sólo limitado por la cantidad de memoria disponible, para la fijación de retornos.

El **RETURN** en una subrutina, produce una bifurcación del programa hacia la instrucción siguiente al **GOSUB** más reciente.

Una subrutina puede contener más de un **RETURN.** La lógica del programa hará llegar al **RETURN** correspondiente según el caso. Ver la sentencia **RETURN** para más detalles.

Las subrutinas pueden aparecer en cualquier punto del programa, pero se recomienda que sean fácilmente distinguibles del programa principal.

En caso de llegarse a la ejecución de una instrucción **RETURN** sin un **GOSUB** previo, se emitirá el mensaje de error: **'RETURN without GOSUB'** (RETURN sin su GOSUB) y se detiene la ejecución del programa.

Asimismo, si dentro de la subrutina se ejecuta una sentencia **CLEAR** o **MAXFILES,** la sentencia **RETURN** no puede devolver el control al programa principal y se genera el mismo error que en el caso anterior.

Ejemplo:

```
10 GOSUB 50
20 GOSUB 50
30 J=1: GOSUB 50
40 J=2: GOSUB 50
50 REM 'SUBRUTINA'
60 PRINT "subrutina";
70 PRINT ":J=";J
80 IF J=2 THEN RETURN 100
90 RETURN
100 END
RUN
subrutina:J= 0
subrutina:J= 0
subrutina:J= 1
subrutina:J= 2
Ok
```

GOTO

GOTO número de línea

Para efectuar una bifurcación incondicional fuera de la secuencia normal del programa, hacia **número de línea.**

número de línea debe ser una línea ejecutable. Luego de bifurcar a **número de línea,** el programa sigue su secuencia normal.

Si **número de línea** no fuera una declaración ejecutable, el programa continuará en la primera línea ejecutable posterior a **número de línea.**

Entre **GO** y **TO** se permite solamente un espacio, pudiendo incluso no existir.

Si se utiliza la sentencia **GOTO** en modo directo, el programa comenzará su ejecución en el **número de línea** especificado. Sin embargo, a diferencia de **RUN, GOTO** no inicializa variables ni cierra archivos.

Ejemplo:

```
10 CLS:GOTO 30
20 PRINT "AMIGA";:GOTO 90
30 PRINT "SOS ";
40 PRINT "MI ";
50 GOTO 20
60 PRINT "COMPUTADORA ";
70 PRINT "TALENT MSX!"
80 PRINT:GOTO 110
90 PRINT ", ";
100 GOTO 60
110 PRINT " - Vos tambien!":END
RUN
SOS MI AMIGA, COMPUTADORA TALENT MSX!

 - Vos tambien!
Ok
```

HEX$

HEX$(expresión entera)

Devuelve una expresión alfanumérica que representa el valor hexadecimal de **expresión entera.**

expresión entera deberá estar en el rango desde -32768 hasta 32767.

Si **expresión entera** es negativa, se usa el complemento a dos. Esto significa que **HEX$ (-n)** es lo mismo que **HEX$(65536 - n).**

La respuesta tiene los ceros no significativos suprimidos.

Ejemplo:

```
10 PRINT "HEX    DEC"
20 FOR J=0 TO 16 STEP 4
30 PRINT RIGHT$("0"+HEX$(J),2);
40 PRINT USING "     ##";J
50 NEXT J
60 END
RUN
```

```
HEX    DEC
00      0
04      4
08      8
0C     12
10     16
Ok
```

IF

IF expresión lógica THEN sentencia [:sentencia...] [ELSE sentencia [:sentencia...]]
IF expresión lógica THEN sentencia [:sentencia...] [ELSE número de línea]
IF expresión lógica THEN número de línea [ELSE número de línea]
IF expresión lógica GOTO número de línea [ELSE sentencia [:sentencia...]]
IF expresión lógica GOTO número de línea [ELSE número de línea]

Para efectuar una **bifurcación condicionada**, fuera de la secuencia normal del programa, dependiendo exclusivamente del **resultado lógico** devuelto por la **expresión lógica** (verdadero o falso).

Si el resultado de la **expresión lógica** es **distinto** de cero (o sea verdadero), las cláusulas **THEN** o **GOTO** son ejecutadas.

THEN puede estar seguido por: un **número de línea**, para bifurcar a dicha instrucción; o por una **sentencia** que se ejecutará. En cambio **GOTO** siempre debe estar seguido por un **número de línea.**

Si el resultado de la **expresión lógica** es cero (falso), entonces las cláusulas posteriores a **THEN** o **GOTO** son ignoradas, y si está presente la cláusula **ELSE**, primero se ejecutan las sentencian que la acompañan, para luego continuar con la ejecución del programa en la próxima línea.

Ejemplo:

```
10 B=2
20 FOR A=1 TO 3
30 PRINT "A=";A;"B=";B;"ENTONCES A=B es: ";
40 IF A=B THEN PRINT "VERDADERO":ELSE PRINT
"FALSO"
```

```
50 NEXT A
60 END
RUN
A= 1 B= 2 ENTONCES A=B es: FALSO
A= 2 B= 2 ENTONCES A=B es: VERDADERO
A= 3 B= 2 ENTONCES A=B es: FALSO
Ok
```

Las instrucciones **IF...THEN...ELSE** pueden estar anidadas y solamente están limitadas por el largo de la línea. Si la declaración no contiene el mismo número de comandos **ELSE** y **THEN**, cada **ELSE** hará juego con el **THEN** más próximo a su izquierda.

Si una instrucción **IF...THEN...ELSE** es seguida por un número de línea en el modo directo, se producirá un mensaje de error: **"Undefined line number"** (número de línea no definido); a menos que una declaración con dicho número de línea haya sido ingresada previamente en el modo indirecto.

Operadores relacionales :

Operador	Tipo de comparación	Ejemplo
=	Igualdad	X = Y
<> ó ><	Desigualdad	X <> Y
<	Menor	X < Y
>	Mayor	X > Y
< = ó =<	Menor o Igual	X < = Y
> = ó =>	Mayor o Igual	X > = Y

Operadores lógicos:

Operador	Tipo de operación	
NOT		
	X	NOT X
	1	Ø
	1	Ø
	Ø	1
	Ø	1
AND		
	X Y	X AND Y
	1 1	1
	1 Ø	Ø
	Ø 1	Ø
	Ø Ø	Ø

OR

X	Y	X OR Y
1	1	1
1	Ø	1
Ø	1	1
Ø	Ø	Ø

XOR

X	Y	X XOR Y
1	1	Ø
1	Ø	1
Ø	1	1
Ø	Ø	Ø

EQV

X	Y	X EQV Y
1	1	1
1	Ø	Ø
Ø	1	Ø
Ø	Ø	1

IMP

X	Y	X IMP Y
1	1	1
1	Ø	Ø
Ø	1	1
Ø	Ø	1

Ejemplo:

```
10 CLS:A=0
20 B=2
30 PRINT ".A..B..C..ENTONCES"
40 FOR C=1 TO 3
50 PRINT A;B;C;
60 IF A=B THEN PRINT " A=B";:IF B=C AND
A=C THEN PRINT " Y A=C" ELSE  PRINT " Y
A<>C" ELSE PRINT" TODOS DISTINTOS"
70 A=A+2:B=A
80 NEXT C
90 END
RUN
.A..B..C..ENTONCES
 0  2  1  TODOS DISTINTOS
 2  2  2  A=B Y A=C
 4  4  3  A=B Y A<>C
Ok
```

INKEY $

Devuelve el caracter cuya tecla fue presionada en el teclado. Si no se presiona ninguna tecla, devuelve un alfanumérico nulo.

Cada vez que se presiona una tecla, el caracter correspondiente a la misma es transferido al buffer del teclado. Cuando se ejecuta **INKEY$**, lee el último dato transferido a dicho buffer.

Las principales diferencias con la instrucción **INPUT** son:

- **No provoca pausa en la ejecución del programa**: ya que en el caso de no pulsarse ninguna tecla, devuelve un caracter "nulo" y sigue la secuencia normal del programa.

Ejemplo:

```
10 PRINT "Prueba INKEY$, con barra espaciadora
termina"
20 J$=INKEY$
40 IF J$=" "THEN END
60 GOTO 10
RUN
```

(En este ejemplo se verificará que hasta tanto no se digite la barra espaciadora el programa seguirá su secuencia normal, o sea sigue imprimiendo el texto de la línea 1Ø)

Si se desea lograr una interrupción del programa la misma se puede lograr consultando si el caracter ingresado es **nulo** y en dicho caso, que vuelva a repetir la instrucción.

Ejemplo:

```
10 PRINT "Prueba INKEY$, con barra espaciadora
termina"
20 J$=INKEY$
30 IF J$="" GOTO 20: REM Consulta caracter nulo
40 IF J$=" "THEN END
60 GOTO 10
RUN
```

(En este ejemplo el programa queda en un ciclo entre las líneas 2Ø y 3Ø hasta que no se presione alguna tecla. Si se presionó una tecla, verifica si es la barra espaciadora, caso contrario vuelve a la línea 1Ø)

- **Los caracteres digitados no son representados en la pantalla:** (no tiene eco), a menos que luego de ingresado el caracter, este sea impreso mediante una instrucción **PRINT.**

Ejemplo:

```
10 PRINT "Prueba INKEY$, con barra espaciadora
termina"
20 J$=INKEY$
30 IF J$="" GOTO 20: REM Consulta caracter nulo
40 IF J$=" "THEN END
50 PRINT J$
60 GOTO 10
RUN
```

(Este ejemplo es similar al anterior. La única diferencia es que imprime J$, que es la variable alfanumérica que almacena el caracter pulsado.)

- **Todos los caracteres digitados son aceptados:** cualquier tecla pulsada es aceptada e ingresada con su código de caracter. La única que no acepta es Ctrl + Stop que provocará la inmediata cancelación del programa.

Ejemplo:

En este ejemplo es interesante probar la digitación de las teclas de función, y las mismas quedarán representadas en la pantalla por las palabras asociadas a cada tecla, con su código en **ASCII** (letra por letra).

```
10 PRINT "Prueba INKEY$, con barra espaciadora
termina"
20 J$=INKEY$
30 IF J$="" GOTO 20: REM Consulta caracter nulo
40 IF J$=" "THEN END
50 PRINT J$;" EL CODIGO ASCII ES:";ASC(J$)
60 GOTO 10
RUN
```

(En este ejemplo se agrega la impresión del código ASCII del caracter pulsado).

INP

INP(número de puerta)

Devuelve el byte leído de la puerta **número de puerta**, el cual debe estar en el rango entre **Ø y 255** .

INP es la función inversa de la declaración **OUT.**

En las declaraciones y funciones arriba mencionadas, la puerta **número de puerta** es manejada mediante caracteres de 16 bits para soportar la capacidad del microprocesador **Z8ØA**, de acceder a las puertas **I/O** (Entrada/Salida) con el par de registros **BC.**

No obstante esto, el **standard MSX** no soporta esos espacios de dirección extendidos a **I/O** y las puertas mayores a **255**, carecen de significado.

Vea el Apéndice E para referencias del mapa de **I/O.**

Ejemplo:

```
10 CLS
20 PRINT "PULSE BARRA ESPACIADORA, HOME, INS,
DEL, Y LOS CURSORES!"
30 OUT 170,(INP(170)AND &HF0)OR 8
40 LOCATE 10,10
50 PRINT RIGHT$("0000000"+BIN$(INP(169)),8)
60 GOTO 30
RUN
PULSE BARRA ESPACIADORA, HOME, INS,  DEL, Y
LOS CURSORES!
```

(Imprime los números binarios correspondientes a cada código de tecla pulsada. Cuando no se pulsa ninguna, el resultado es 1111111. Detener con [CTRL] + [STOP]).

INPUT

INPUT ["mensaje alfanumérico";] variable [, variable ,...]

Para habilitar el ingreso de datos numéricos o alfanuméricos desde el teclado durante la ejecución de un programa.

Cuando se encuentra una declaración **INPUT**, la ejecución del programa hace una pausa y aparece un signo de interrogación (?) para indicar que el programa espera el ingreso de un dato.

Si se incluye un **mensaje alfanumérico** , este es impreso antes del signo de interrogación.

Cuando se pulsa la tecla [RETURN], la información digitada se transfiere a la variable especificada. Si sólo se pulsa [RETURN], sin digitar nada previamente, el valor de la variable permanece inalterado.

El o los datos ingresados son asignados a las variables en órden correlativo, y, por lo tanto, el número de datos provisto debe ser igual al de las variables y los mismos estarán separados por comas.

Las **variables** pueden ser numéricas o alfanuméricas (inclusive elementos de una matriz).

Los datos ingresados con el **INPUT** deben coincidir con los tipos de variables especificados por los nombres de las mismas. (Los datos alfanuméricos que ingresan en una declaración **INPUT** no requieren hallarse entre comillas, a menos que contengan comas)

Responder al **INPUT** con un tipo de variable equivocado (alfanumérica en lugar de numérica, etc.) provocará un mensaje de atención por pantalla: **"?Redo from start"**(repita desde el principio) y queda a la espera de la respuesta correcta.

Ejemplo:

```
10 INPUT "INGRESAR: A y B";A,B
20 PRINT "EL RESULTADO ES:";A+B
30 END
RUN
```

```
INGRESAR: A y B? 10,PEPE
?Redo from start
INGRESAR: A y B? 10  20
EL RESULTADO ES: 30
Ok
```

Responder a **INPUT** con más datos que las variables indicadas provoca el mensaje de atención por pantalla: **"?Extra ignored"** (dato extra ignorado) y se continúa con la ejecución secuencial del programa.

Ejemplo:

```
10 INPUT "INGRESAR: A y B";A,B
20 PRINT "EL RESULTADO ES:";A+B
30 END
RUN
INGRESAR: A y B? 10,20,30
?Extra ignored
EL RESULTADO ES: 30
Ok
```

Si se responde a un **INPUT** con menos ítems que los previstos, provoca la impresión de dos signos de pregunta y espera que ingrese el/los datos faltantes.

Ejemplo:

```
10 CLS
20 INPUT "INGRESA TU APELLIDO, NOMBRE:";A$,N$
30 PRINT"HOLA ";N$;", SOY TU TALENT MSX."
40 END
RUN
INGRESA TU APELLIDO, NOMBRE:? PEREZ
??CARLOS
HOLA CARLOS, SOY TU TALENT MSX.
Ok
```

Se puede pasar de una instrucción **INPUT** al nivel de comando tipeando [CONTROL] + [C] o [CTRL] + [STOP].

Para retornar del nivel de comando a la instrucción **INPUT**, se debe tipear **CONT** e inmediatamente se reasume la ejecución en el mismo punto en que fue interrumpida, o sea desde el **INPUT**.

Si se ejecuta una sentencia **INPUT** en modo gráfico (**SCREEN 2 ó 3**), el sistema retornará automáticamente al modo texto (**SCREEN Ø**.

INPUT

INPUT# número de archivo , variable [, variable.....]

Para leer los datos requeridos desde el canal correspondiente a **número de archivo** y asignárselo a las **variables** especificadas.

El archivo especificado debe ser previamente abierto con la sentencia **OPEN** en el modo de ingreso (**INPUT**).

Los datos del archivo deberán coincidir en tipo con los de cada una de las **variables** e ingresan tal como aparecerían si los mismos fueran tipeados por consola. Se ignoran todos los **return** y **line feed** que acompañen a la información leída.

Si el primer caracter de un dato alfanumérico leído es una comilla, la cadena consistirá en todos los caracteres leídos entre la primera y segunda comilla. Por consiguiente el contenido de una cadena entre comillas, no podrá contener otra comilla como caracter.

Caso contrario el mismo requerirá terminar en una coma, **return, line feed,** o después de haberse leído 255 caracteres.

return es el caracter **ASCII 13** y **line feed** es el caracter **ASCII 10**

Si se alcanza el fin del archivo cuando se hace el **INPUT #** de una variable numérica o alfanumérica, se da por cumplida la instrucción.

Ejemplo:

```
10 OPEN"CAS:DATA" FOR INPUT AS #1
20 INPUT #1,A,B
30 PRINT "DATA A=";A
40 PRINT "DATA B=";B
50 END
RUN
```

(Se leen los datos grabados en el cassette bajo el nombre de DATA y luego los imprime. DATA debería tener almacenadas dos variables numéricas).

```
Ok
```

INPUT $

INPUT$ (expresión entera, [#] número de archivo)
INPUT$ (expresión entera)

Para leer un dato alfanumérico cuya longitud está indicada por expresión entera desde el archivo indicado por **número de archivo.**

número de archivo es el número con el cual fue abierto el archivo. Si éste se omite, se supondrá que es el teclado pero no se visualizan los caracteres digitados.

Una vez que se ingresaron el total de caracteres indicado por expresión entera, sigue la ejecución del programa sin que se requiera pulsar la tecla [RETURN].

Esta función lee cualquier codigo de caracter excepto la secuencia [CTRL] + [STOP]. Dentro de los caracteres aceptados se encuentran CR (return, código ASCII 13) y LF (line feed, código ASCII 10)

Ejemplo:

```
10 'PASE : MSX + tecla "ESC"
20 P$="MSX"+CHR$(27)
30 PRINT "SU PASE?";
40 A$=INPUT$(4)
50 PRINT A$
60 IF A$<>P$ THEN PRINT"NO ESTA AUTORIZADO!":GOTO 30
70 PRINT
80 BEEP
90 PRINT "BIENVENIDO AL MUNDO DE TALENT MSX"
100 END
RUN
SU PASE?ABCD
NO ESTA AUTORIZADO!
SU PASE?MSX
BIENVENIDO AL MUNDO DE TALENT MSX
Ok
```

INSTR

INSTR ([expresión entera,] expresión alfanumérica 1, expresión alfanumérica 2)

Devuelve la posición del primer caracter de la primera coincidencia de **expresión alfanumérica 2** con **expresión alfanumérica 1.**

El opcional **expresión entera** determina la posición del comienzo de la búsqueda dentro de **expresión alfanumérica 1** y deberá estar dentro del rango de **1** a **255.**

Por ejemplo si ponemos **INSTR(n,X$,Y$)**

INSTR devolverá **Ø** si:

- No se puede encontrar coincidencia con Y$
- **LEN(Y$)** mayor que **LEN(X$)**
- X$ es nulo
- X$ e Y$ son nulas
- **n** mayor que **LEN(X$)**

Ejemplo:

```
10 '*** PIANO ***
20 CLS:PRINT "R=DO":LPRINT"TOQUE EL SIGUIENTE
TEMA:","TTUTPO","TTUTJP","TTGS","P
POU","DDSPJP"
30 TB$="QWERTYUIOP[]ASDFGHJKL;''ZXCVBNM,./ "
40 PLAY"L8S7M1000"
50 IN$=INKEY$
60 IF IN$="" THEN GOTO 50
70 I=INSTR(TB$,IN$)
80 IF I>0 THEN I=I+32
90 PLAY"N=I;"
100 GOTO 50
RUN
R=DO
TOQUE EL SIGUIENTE TEMA:
TTUTPO          TTUTJP
TTGS            PPOU
DDSPJP
```

(Si sigue las instrucciones podrá tocar un tema conocido... Detener el programa con [CTRL] + [STOP])

INT

INT(expresión numérica)

Devuelve la menor parte entera de **expresión numérica** (fracción truncada).

La diferencia fundamental entre **INT** y **FIX** es que **INT** devuelve el número próximo inferior para **expresión numérica** negativa.

Ejemplo:

```
10 PRINT "  X        FIX(X)  INT(X)"
20 FOR J=-1.2345 TO 1.2345 STEP .5
30 PRINT USING "##.####";J;
40 B=FIX(J)
50 PRINT USING "######";B;
60 B=INT(J)
70 PRINT USING "######";B
80 NEXT J
90 END
RUN
  X        FIX(X)  INT(X)
-1.2345     -1     -2
-0.7345      0     -1
-0.2345      0     -1
 0.2655      0      0
 0.7655      0      0
Ok
```

INTERVAL ON/OFF/STOP

Para activar o desactivar la ejecución de una interrupción que genera automáticamente la máquina cada lapso determinado de tiempo, programable.

Una declaración **INTERVAL ON** debe ser ejecutada para activar la definición de un intervalo de tiempo.

Después del **INTERVAL ON,** si se especifica un número de línea en el **ON INTERVAL GOSUB,** cada vez que el **BASIC** comienza una

nueva instrucción, verificará el intervalo de tiempo transcurrido y cuando encuentre que se cumplió el tiempo del intervalo ejecutará el **GOSUB** hasta el número de línea especificada .

Si el **INTERVAL OFF** ha sido ejecutado, no habrá definición y el hecho no será recordado, aún si esto sucede.

Habiéndose cumplido un **INTERVAL STOP**, no habrá definición, pero si ocurre la interrupción del temporizador, ésta será recordada y tendrá lugar una activación inmediata.

Ver la instrucción **ON INTERVAL GOSUB** para más detalles.

Ejemplo:

```
10 CLS:ON INTERVAL=30 GOSUB 150
20 SCREEN 0:WIDTH 37:COLOR 15,4,4
30 INTERVAL ON:KEYOFF:DEFINT A-Z
40 PRINT "CALCULO DE NUMEROS PRIMOS":PRINT
"RANGO 3-100"
50 FOR I=3 TO 100
60 A=1
70 A=A+1:IF I=(I\A)*A THEN 100
80 IF A<I-1 THEN 70
90 PRINT "[";I;"]-";:C=C+1
100 NEXT I:PRINT:PRINT "TOTAL DE NUMEROS
PRIMOS:";C
110 END
120 '
130 ' RUTINA BIP
140 '
150 BEEP:PRINT TAB(29);" ...Bip!":RETURN
RUN
```

(Calcula los números primos en el rango indicado y cada medio segundo emite un "Bip" con su correspondiente impresión.)

KEY

KEY expresión entera, expresión alfanumérica

Para asignar el contenido de **expresión alfanumérica** a la tecla de función especificada por **expresión entera.**

expresión entera deberá estar en el rango de 1 a 10, que corresponden a las teclas de función.

La longitud de **expresión alfanumérica** deberá estar dentro de los 15 caracteres (aunque en pantalla solo se visualizarán los 7 primeros). Todos los caracteres extras se ignoran.

Las definiciones asignadas a las 5 primeras teclas de función aparecerán desplegadas en el renglón 24 de la pantalla, y presionando la tecla de mayúscula, se visualizarán las correspondientes a las últimas 5 teclas de función.

Cuando se coloca un código de control en **expresión alfanumérica**, deberá estar precedido por el signo más **(+)** y la función **CHR$**.

Ejemplo:

```
KEY 1,"NEW"+CHR$(13)+"CLS"+CHR$(13)
```

En el renglón 24 de la pantalla aparecerá ahora visualizado: **"NEW CLS"**, verifique ahora que ocurre al tipear la tecla de función 1.

La operación efectuada equivale a escribir: **"NEW"** [RETURN] y **"CLS"** [RETURN].

KEY LIST

Para listar por pantalla las 10 constantes asignadas a cada una de las teclas de función.

La posición en la lista refleja la asignación de las teclas.

Los caracteres de control asignados a cada una de las constantes por tecla de función son convertidos en espacios en el momento de la impresión.

Ejemplo:

```
KEY LIST [RETURN]
color
auto
goto
list
run
```

```
color 15,4,7
cload"
cont
list.
run
Ok
```

"color" se opera con la tecla **"f1"**, **"auto"** con la **"f2"**, **"goto"** con la **"f3"** y así sucesivamente.

KEY ON/OFF

Para activar o desactivar sobre la pantalla, en su línea 24, la impresión de las constantes asignadas a cada una de las teclas de función.

Mientras estén activadas las teclas de función mediante el **KEY ON**, la pantalla solo tendrá disponibles para el usuario **23** renglones; en cambio al desactivarse con el **KEY OFF**, se dispondrán de los **24** renglones.

Ejemplo:

```
KEY OFF [RETURN]
Ok
```

(Desaparecen de la pantalla el listado de funciones).

```
KEY OFF[RETURN]
Ok
```

(Reaparece la informacion en pantalla):

color auto goto list run

KEY (n) ON/OFF/STOP

KEY (expresión entera) ON / OFF / STOP

Para activar o desactivar las interrupciones en un programa **BASIC** debido a la detección del pulsado de la tecla de funcion especificadas

Una declaración **KEY(n) ON** deberá ser ejecutada para activar la definición de cada una de las teclas de función, siendo **n** la **expresión entera.**

Después de una declaración **KEY(n) ON**, si se especificara un número de línea en **ON KEY GOSUB**, cada vez que un programa **BASIC** comience una nueva sentencia, verificará si dicha tecla ha sido oprimida, en cuyo caso se ejecutará el **GOSUB** al número de línea consignado.

Si ha sido ejecutado un comando **KEY(n) OFF**, no habrá definición y el hecho no será memorizado aunque tenga lugar.

En caso de cumplirse un comando **KEY(n) STOP** no habrá definición, pero si la tecla indicada es seleccionada, ésta será memorizada, de manera que la definición tendrá lugar inmediatamente después de ser ejecutada **KEY(n) ON.**

KEY(n) ON no tiene efecto aunque el valor de la tecla de función sea mostrado al pie de la pantalla.

Ejemplo:

```
10 ON KEY GOSUB 60,70,80,90,100,110,120,130,
140,150
20 FOR J=1 TO 10
30 KEY(J)ON
40 NEXT J
50 GOTO 50
60 N=1:GOTO 160
70 N=2:GOTO 160
80 N=3:GOTO 160
90 N=4:GOTO 160
100 N=5:GOTO 160
110 N=6:GOTO 160
120 N=7:GOTO 160
130 N=8:GOTO 160
140 N=9:GOTO 160
150 N=10:GOTO 160
160 PRINT USING " TECLA SELECCIONADA: F-##";N
170 RETURN
RUN
```

(Se activa la interrupción por tecla de función y luego el programa queda encerrado en el ciclo de la línea **5Ø** Sin embargo, cada vez que se pulsa una tecla de función vemos que bifurca a la subrutina **16Ø** y si se hubieran pulsado las teclas **f1**, **f3** ó **f1Ø** aparecería:)

```
TECLA SELECCIONADA: F- 1
TECLA SELECCIONADA: F- 3
```

(Detener con [CTRL] + [STOP]).

LEFT $

LEFT$(expresión alfanumérica,cantidad)

Devuelve una contante alfanumérica formada con los primeros **cantidad** caracteres de **expresión alfanumérica.**

cantidad deberá estar en el rango de **Ø a 255.**

Si **cantidad** es mayor que **LEN(expresión alfanumérica)** se obtendrá el contenido de **expresión alfanumérica** completo, en cambio si **cantidad=Ø** retornará un alfanumérico nulo (de largo cero).

El encabezamiento de gráficos (caracter ASCII 1) de un símbolo gráfico se cuenta como un caracter. Esto significa que un símbolo gráfico se cuenta como dos caracteres.

Ejemplo:

```
10 INPUT A$
20 FOR J=1 TO LEN(A$)
30 PRINT LEFT$(A$,J)
40 NEXT J
50 END
RUN
?TALENT
T
TA
TAL
TALE
TALEN
TALENT
Ok
```

LEN

LEN (expresión alfanumérica)

Efectúa el recuento de caracteres en **expresión alfanumérica,** incluyéndose en el conteo los caracteres no imprimibles y los espacios (caracter ASCII 32).

Para los caracteres gráficos se cuenta también el encabezamiento de

gráficos (caracter código ASCII 1). Luego, los caracteres graficos dan longitud 2 por cada caracter.

Ejemplo:

```
10 A$="LA COMPUTADORA TALENT-MSX"
20 PRINT A$;" TIENE:";LEN(A$);" LETRAS:
30 END
RUN
LA COMPUTADORA TALENT-MSX TIENE: 25  LETRAS
Ok
```

LET

[LET] variable = expresión

Para asignar el valor de una expresión a una variable.

Es importante hacer notar que la palabra LET es opcional ya que el signo igual (=) es suficiente cuando se asigna el valor de una expresión a una variable.

No es posible asignar el valor de una expresión alfanumérica a una variable numérica o el valor de una expresión numérica a una variable alfanumérica.

Ejemplo:

```
10 LET A=10
20 PRINT A
30 A=25
40 PRINT A
50 LET A=48
60 PRINT A
70 END
RUN
 10
 25
 48
Ok
```

LINE

LINE [(X1,Y1) ó STEP(X1,Y1)] - (X2,Y2) ó STEP(X2,Y2) [, **código de color**] [, **tipo dibujo**]

Para dibujar una línea conectando los puntos especificados por las coordenadas **(X1,Y1)** y **(X2,Y2).**

(X1,Y1) son las coordenadas del punto inicial y **(X2,Y2)** son las coordenadas del punto final de la línea tomando como referencia la esquina superior izquierda.

Si se especifica **STEP(X1,Y1)** ó **STEP(X2,Y2),** el punto quedará determinado en forma relativa al último punto impreso.

Si se omite **(X1,Y1)** se supone que la línea conecta el punto **(X2,Y2)** con el último punto dibujado en la pantalla. No se pueden omitir las coordenadas **(X2,Y2).**

Para más detalles del **(X1,Y1) etc.**, ver la instrucción **PUT SPRITE,** y para **código color** ver la tabla correspondiente a la instrucción **COLOR.**

En **tipo dibujo** si se indica **'B'** dibujará el perímetro de un rectángulo y si se indica **'BF'** no solo dibujará el perímetro del rectángulo sino que pintará la superficie del mismo.

Luego de ejecutar la sentencia **LINE,** el último punto referenciado (que se utiliza para coordenadas relativas) es **(X2,Y2).**

Ejemplo:

```
10 SCREEN 2:DEFINT A-Z
20 CLS:OPEN"GRP:"FOR OUTPUT AS#1
30 FOR C=10 TO 15
40 FOR J=0 TO 95 STEP 3
50 LINE(131-J,95-J)-(131+J,95+J),C,B
60 NEXT J
70 NEXT C
80 LINE(81,45)-(181,145),1,BF
90 PSET(96,60):PRINT#1,"TALENT-MSX"
100 FOR J=0 TO 2000
110 NEXT J
120 END
RUN
```

(Verá otra demostración más de la capacidad de gráficos de la Talent-MSX)

LINE INPUT

LINE INPUT ["mensaje alfanumérico";] variable alfanumérica

Para ingresar una línea entera (hasta 254 caracteres) a una variable alfanumérica, sin el uso de separadores de campo.

El **mensaje alfanumérico** es una serie de caracteres que se imprime en la pantalla antes que el ingreso sea aceptado.

No se imprimirá un signo de interrogación a menos que sea parte del **mensaje alfanumérico.**

Todos los datos ingresados desde el final del **mensaje alfanumérico** hasta el [RETURN] es asignado a la **variable alfanumérica.**

Si sólo se pulsa [RETURN] sin haber digitado nada, se le asigna el alfanumérico nulo (**""**) a **variable alfanumérica.**

Para salir del **LINE INPUT** se deberá tipear [CTRL] + [C] o [CTRL] + [STOP], retornando entonces al nivel comando; tipeando **CONT** se continúa con la secuencia lógica del programa a partir del **LINE INPUT.**

LINE INPUT no regresa al modo texto si se llama desde una pantalla gráfica, y no se asigna correctamente los datos digitados.

Ejemplo:

```
10 PRINT "PARA IMPRIMIR";CHR$(34);"Y";CHR$(44)
20 PRINT "POR MEDIO DEL 'LINE INPUT'"
30 LINE INPUT A$
40 PRINT "A$= ";A$
50 END
RUN
PARA IMPRIMIR"Y,
POR MEDIO DEL 'LINE INPUT'
"TALENT - MSX"
A$= "TALENT - MSX"
Ok
```

LINE INPUT

LINE INPUT# número de archivo; variable alfanumérica

Para leer una línea entera (hasta un máximo de 254 caracteres) sin delimitadores, desde el archivo secuencial abierto según **número de archivo** hacia la **variable alfanumérica.**

LINE INPUT# lee todos los caracteres en el archivo secuencial hasta el primer **return**, saltando la secuencia **return / line feed** y el próximo **LINE INPUT#** leerá todos los caracteres hasta el próximo **return.**

En caso de encontrar una secuencia **return / line feed,** ésta sí será preservada y serán devueltos como parte de la **variable alfanumérica.**

LINE INPUT # es especialmente útil si cada línea de un archivo ha sido partida en campos, o si un programa en **BASIC** grabado en modo **ASCII** debe ser leído como dato por otro programa.

return es el caracter código ASCII 13 y line feed el caracter código ASCII 10.

Ejemplo:

```
10 OPEN"CAS:DATA" FOR INPUT AS #1
20 LINE INPUT#1,A$
30 PRINT A$
40 CLOSE
50 END
RUN
```

(Se lee el archivo **DATA** almacenado en cassette y se ingresa cada línea completa del mismo, asignándola a la variable alfanumérica **A$.**)

LIST

LIST [número de línea inicial] [- número de línea final]

Para listar por pantalla las líneas de un programa que está en memoria, en forma parcial o total.

Si ambos parámetros **número de línea** son omitidos, el programa será listado comenzando por el número de línea más bajo.

Especificando únicamente **número de línea inicial** sólo se listará dicha línea.

Cuando **número de línea inicial-** son especificados, dicha línea y todas las líneas de numeración más alta serán listadas.

Al indicar **-número de línea final** serán listadas todas las líneas desde el principio del programa hasta esa línea.

Si ambos parámetros **números de línea** son detallados, las instrucciones con rango desde **número de línea inicial** hasta **número de línea final** serán listadas.

Para dar por finalizado el listado se deberán tipear las teclas [CTRL] + [STOP] pero para suspender momentáneamente el mismo se debe oprimir la tecla [STOP], y para continuarlo se digitará nuevamente la misma tecla.

Por último, utilizando LIST. se obtiene el listado de la última línea ejecutada o ingresada y el cursor queda sobre dicha línea, para permitir su corrección. Es ideal para cuando se comete un error de ejecución, ya que basta digitar LIST. [RETURN] para que la línea en cuestión quede impresa y lista para corrección.

Ejemplo:

```
10 A=3
20 B=6
30 C=A+B
40 PRINT A,B,C
50 END
LIST
10 A=3
20 B=6
30 C=A+B
40 PRINT A,B,C
50 END
Ok
LIST 20
20 B=6
Ok
LIST -30
10 A=3
20 B=6
30 C=A+B
Ok
LIST 20-40
20 B=6
30 C=A+B
40 PRINT A,B,C
Ok
```

LLIST

LLIST [número de línea inicial] - [número de línea final]

Para listar por impresora las líneas de un programa que está en memoria, en forma parcial o total.

(Ver el comando **LIST** para más detalles de los parámetros).

LOAD

LOAD " [: nombre de archivo]" [, R]

Para cargar un programa **BASIC** desde el dispositivo indicado en **dispositivo.**

Antes de comenzar la lectura del programa el **LOAD** cierra todos los archivos abiertos y borra el programa que se encuentre en la memoria.

Si **dispositivo** es **"CAS:"** el programa debe estar almacenado en formato **ASCII**, o sea, mediante la instrucción **SAVE** (**NO CSAVE**).

La diferencia fundamental con **CLOAD** es que en este último sólo se permite utilizar cassettes, y que el archivo utilizado está en formato "imagen de memoria", incompatible con el formato **ASCII.**

Con la opción **"R"** todos los archivos de datos permanencerán abiertos y comenzará la ejecución del programa una vez cargado en la memoria.

Si se omite **nombre de archivo,** el próximo programa que se encuentre en la cinta, que debería ser un archivo **ASCII,** será cargado directamente.

Si se lee un **CTRL + Z** , el mismo será tratado como un **fin-de-archivo** (end-of-file).

Pueden utilizarse otros dispositivos para cargar programas en formato **ASCII,** como ser drive de diskettes o interfase de comunicaciones **RS232.**

Ejemplo:

LOAD "CAS:TEST"
Found:TEST
Ok
(Se carga desde cassette el programa TEST, previamente grabado en formato ASCII)

LOCATE

LOCATE [número de columna] [,número de fila] [, activador de impresión del cursor]

Para posicionar el cursor en la pantalla, donde **número de columna** determina la columna, y **número de fila,** la fila. Columna Ø, fila Ø es el borde superior izquierdo de la pantalla.

número de columna va del rango **Ø a 39.**

Si **número de columna** excede el ancho máximo especificado por la sentencia **WIDTH,** se toma el valor máximo permitido por esta sentencia.
Cuando se omite **número de columna,** se asume el número de columna previo.

número de fila va del rango de **Ø a 23.**

Si **número de fila** excede la última posición, se asume dicha fila (la número **23**).
La última fila que se imprime es la **22** si están impresas las teclas de función, y **23** si no lo están.
Cuando no se especifica **número de fila,** se asume el número de columna previo.

activador de impresión del cursor puede utilizarse solamente en los modos de texto (**SCREEN Ø y 1**):

Ø: desactiva la impresión del cursor
1: activa la impresión del cursor

Cuando el activador está en Ø, no aparece ningún cursor en la pantalla excepto cuando el sistema espera que ingrese un dato por teclado. Cuando está en 1, se imprime el cursor. El valor inicial es Ø.

Ejemplo:

```
10 A$="ABCDEFGHIJKLMN"
20 CLS
30 FOR J=1 TO 15
40 LOCATE J,J-1
50 PRINT LEFT$(A$,J)
60 NEXT J
70 END
RUN
 A
  AB
   ABC
    ABCD
     ABCDE
      ABCDEF
       ABCDEFG
        ABCDEFGH
         ABCDEFGHI
          ABCDEFGHIJ
           ABCDEFGHIJK
            ABCDEFGHIJKL
             ABCDEFGHIJKLM
              ABCDEFGHIJKLMN
               ABCDEFGHIJKLMN

Ok
```

LOG

LOG(expresión numérica)

Permite calcular el logaritmo natural o logaritmo base e de **expresión numérica**, que deberá ser mayor que cero.

El resultado que devuelve es siempre un número real de doble precisión, sin interesar el tipo que sea **expresión numérica.**

Ejemplo:

```
10 FOR J=10 TO 50 STEP 10
20 PRINT "LOG(";J;")=";LOG(J)
30 NEXT J
40 END
RUN
```

```
LOG( 10 )= 2.302585092994
LOG( 20 )= 2.995732273554
LOG( 30 )= 3.4011973816622
LOG( 40 )= 3.6888794541139
LOG( 50 )= 3.912023005428
Ok
```

LPOS

LPOS(expresión)

Devuelve la dirección en que se encuentra posicionada la cabeza impresora dentro del buffer de impresión (en la memoria).

No da necesariamente la posición física de la cabeza impresora, ya que pueden existir caracteres de control que serán interpretados como un avance de la misma.

expresión es un argumento simbólico.

Ejemplo:

```
10 PRINT "EN 00:";LPOS(X)
20 LPRINT "ABCDEFG";
30 PRINT "EN 30:";LPOS(X)
40 LPRINT "HIJKLNOP";
50 PRINT "EN 50:";LPOS(X)
60 LPRINT "QRS"
70 PRINT "EN 70:";LPOS(X)
80 END
RUN
EN 00: 0
EN 30: 7
EN 50: 15
EN 70: 0
Ok
```

(Requiere tener conectada la impresora).

LPRINT/LPRINT USING

Para imprimir datos con la impresora (para más detalles ver las declaraciones **PRINT** y **PRINT USING**)

Ejemplo:

```
10 PRINT "INGRESE CUALQUIER FRASE"
20 LINE INPUT A$
30 PRINT A$
40 LPRINT A$
50 END
RUN
INGRESE CUALQUIER FRASE
TALENT MSX [RETURN]        (su ingreso por teclado)
TALENT MSX                 (en la pantalla)
TALENT MSX                 (por la impresora)
Ok
```

MAXFILES

MAXFILES = expresión entera

Para indicar el número máximo de archivos que se pueden abrir simultáneamente.

expresión entera debe estar dentro del rango desde **Ø a 6.**

Cuando se ejecuta **'MAXFILES = Ø** sólo pueden utilizarse **SAVE** y **LOAD.**

El valor normal asignado es **1.**

Al igual que la sentencia **CLEAR, MAXFILES** inicializa todas las variables.

Las variables numéricas se inicialan en cero.
Las variables alfanuméricas se inicialan en nulo ("").
Todos los archivos son cerrados.
Todos las instrucciones que comienzan con **DEF** (**DEF FN, DEF USR, DEFINT, DEFSNG, DEFDBL,** y **DEFSTR**) son canceladas.

Se eliminan todas las variables con subíndice.
Los ciclos **FOR - NEXT** son discontinuados.
RETURN no devuelve el control al programa principal.

Cuando se ejecuta la sentencia **MAXFILES**, se reserva en la memoria tantos bloques de control de archivo como indique **expresión entera + 1.** Se reservan **267** bytes por cada bloque de control de archivo.

Ejemplo:

```
10 MAXFILES=3
20 OPEN"CAS:DATA1" FOR OUTPUT AS#1
30 OPEN"CRT:DATA2" FOR OUTPUT AS#2
40 OPEN"LPT:DATA3" FOR OUTPUT AS#3
50 ...
```

(Se declara que la máxima cantidad de archivos disponibles es 3. En este caso,. los tres archivos seleccionados son: **CAS** (cassette), **CRT** (pantalla) y **LPT** (impresora))

MERGE

MERGE "Dispositivo : [nombre de archivo] "

Para **intercalar** las líneas de un programa grabado en formato **ASCII,** al programa que se encuentra en memoria.

Cualquier línea del archivo que está siendo intercalado, si tiene el mismo número de línea que la del programa que está en memoria, la del archivo reemplazará a su correspondiente en la memoria.

Después del comando **MERGE** (intercalar), el programa "mezclado" (intercalado) reside en memoria y el **BASIC** devuelve el nivel comando.

Si se omite **nombre de archivo** el próximo archivo de programa que encuentre en la cinta, que debería ser un archivo **ASCII,** será intercalado.

Si se detecta **Control + Z** en la lectura, será tratado como **fin-de-archivo.**

Ejemplo: Tenemos el siguiente programa en memoria:

```
10 CLS: PRINT "INICIO"
15 PRINT "ESTA ES UNA PRUEBA"
20 PLAY"A"
```

Y en cassette tenemos grabado el programa **MEZCLA** con las siguientes líneas:

```
5 REM MEZCLA
10 SCREEN 0: PRINT "COMIENZO"
15 PRINT "ESTA ES UNA PRUEBA"
20 FOR I=1 TO 10
30 PRINT I;
40 NEXT
```

Tipeamos:

```
MERGE"CAS:MEZCLA"
Found:MEZCLA
Ok
```

Ahora listamos nuevamente el programa y vemos:

```
5 REM MEZCLA
10 SCREEN 0:PRINT"COMIENZO"
15 PRINT"ESTA ES UNA PRUEBA"
20 FOR I=1 TO 10
30 PRINT I;
40 NEXT
Ok
```

(Ambos programas fueron "mezclados" en uno solo)

MID$

APLICACION 1:

MID $ (expresión alfanumérica , caracter inicial [, cantidad])

Devuelve una porción de **expresión alfanumérica** que comienza en **caracter inicial** y posee **cantidad** caracteres de longitud.

caracter inicial es una expresión entera cuyo rango va desde **1** a **255.**

Si **caracter inicial** es mayor que **LEN(expresión alfanumérica)**, devuelve un alfanumérico nulo.

cantidad es una expresión entera cuyo rango va desde **Ø** a **255**.
Cuando **cantidad** se omite, **MID$** devuelve el caracter especificado en **caracter inicial** y todos los que le siguen a la derecha.
Si **cantidad=Ø**, **MID$** devuelve el alfanumérico nulo ("").

El encabezamiento de gráficos se cuenta como un caracter (código ASCII 1). Esto significa que los caracteres gráficos ocupan 2 posiciones.

Ejemplo:

```
10 CLS:SCREEN 1:WIDTH 28
20 READ A$
30 FOR I=1 TO LEN(A$)
40 LOCATE 0,10:PRINT MID$(A$,I,28):BEEP:FOR
J=1 TO 5:NEXT J
50 NEXT I
60 END
70 DATA LA COMPUTADORA TALENT MSX DPC-200
POSEE UNA GRAN CAPACIDAD DE GRAFICOS Y
UN BASIC MUY FLEXIBLE. APROVECHELO!      FIN
TRANSMISION.
RUN
```

(Verá la impresión del mensaje contenido en **DATA**, pero simulando el formato de un télex)

APLICACION 2:
MID $ (variable alfanumérica , caracter inicial !, cantidad !) = expresión alfanumérica

Para reemplazar una parte de una variable alfanumérica en otra.

A partir de la posición **caracter inicial** de **variable alfanumérica**, se reemplazarán **cantidad** caracteres, tomados de **expresión alfanumérica**, a partir de la posición **1**.

Aunque **cantidad** sea omitida o incluída, el reemplazo de caracteres nunca excederá del largo original de la **variable alfanumérica**.

caracter inicial es una expresión entera cuyo rango va desde 1 a 255.
caracter inicial no debe exceder el número de caracteres asignados a **variable alfanumérica**.

cantidad es una expresión entera cuyo rango va desde Ø a 255. Cuando **cantidad** se omite o excede el número de caracteres de **expresión alfanumérica**, el número de caracteres que se reemplazan es igual al número que contiene **expresión alfanumérica.**

Cuando el número de caracteres a ser reemplazados es menor que el número de caracteres de **expresión alfanumérica**, los caracteres de la variable son reemplazados por el número de caracteres correspondientes en la **expresión alfanumérica**, comenzando por el primer caracter.

Cuando el valor (**caracter inicial+cantidad-1**) excede el número de caracteres de la **variable alfanumérica**, los caracteres extra son ignorados.

Los caracteres gráficos ocupan dos posiciones, ya que se cuenta el caracter encabezamiento de gráficos (código ASCII 1).

Ejemplo:

```
10 A$="abcdefg"
20 B$="h swearty"
30 PRINT A$;" ";B$
40 PRINT
50 FOR J=1 TO 7
60 C$=A$
70 MID$(C$,J,3)=B$
80 PRINT C$," ";J
90 NEXT J
100 END
RUN
abcdefg h swearty

h sdefg          1
ah sefg          2
abh sfg          3
abch sg          4
abcdh s          5
abcdeh           6
abcdefh          7
Ok
```

MOTOR

MOTOR ON
MOTOR OFF
MOTOR

Para cambiar el estado del interruptor del motor del grabador de cassettes.

Cuando no se provean argumentos, actúa sobre la llave interruptora del grabador de cassettes, cambiando al estado opuesto al actual. Caso contrario, activa (ON) o desactiva (OFF) el motor del mismo.

El estado normal (cuando recién se enciende la computadora) es OFF.

Ejemplo:

```
MOTOR ON
```
(Si tiene un grabador conectado, verificará que si pulsa la tecla **avance (PLAY)** del mismo, la cinta avanza)
```
Ok
MOTOR OFF
```
(En cambio ahora verá que si repite la operación, la cinta no avanza. Siempre se supone que su grabador posee la conección para REMOTE).
```
Ok
```

NEW

Para borrar la totalidad de la memoria de trabajo y eliminar todas las variables.

Cuando se ejecuta la sentencia NEW, todos los archivos se cierran, se elimina el programa de memoria, así como las variables con subíndice, y se cancelan todas las declaraciones que comienzan con DEF (DEF FN, DEF USR, DEFINT, DEFSNG, DEFDBL o DEFSTR) son canceladas, incluyendo las ingresadas en modo directo.

Ejemplo:

```
10 A$="ABC12345+*?="
20 PRINT MID$(A$,4,7)
30 END
RUN
12345+*
Ok
PRINT A$
ABC12345+*?=
Ok
NEW
Ok
list
Ok
run
Ok
PRINT A$
Ok
```

NEXT

NEXT [variable contador [,variable contador...]]

Vea **RESUME** para el uso de **NEXT** con esa instrucción.

La instrucción **NEXT** acompaña siempre a la instrucción **FOR** cuya **variable contador** coincide, formando un ciclo.

Cuando se indican más de una **variable contador** en la misma sentencia **NEXT**, se forman ciclos con el correspondiente número de sentencias **FOR**. En este caso, las **variable contador** en la sentencia **NEXT** deben estar ubicadas de tal forma que la primera corresponda con la instrucción **FOR** más cercana, la segunda con la siguiente, etc.

Cuando se omite **variable contador**, la sentencia **NEXT** hace juego con el **FOR** más cercano.

Ejemplo:

```
10 CLS:PRINT "FUNCION Y=X^2"
20 FOR I=0 TO 5
30 PRINT "X=";I;TAB(10);"Y=";I^2
40 NEXT
50 PRINT:PRINT"FUNCION Y=X^3"
60 FOR I=0 TO 5
70 PRINT "X=";I;TAB(10);"Y=";I^3
80 NEXT I
90 FOR I=1 TO 3
100 FOR J=1 TO 2
110 PRINT I;J:NEXT J,I
120 END
RUN
FUNCION Y=X^2
X= 0        Y= 0
X= 1        Y= 1
X= 2        Y= 4
X= 3        Y= 9
X= 4        Y= 16
X= 5        Y= 25

FUNCION Y=X^3
X= 0        Y= 0
X= 1        Y= 1
X= 2        Y= 8
X= 3        Y= 27
X= 4        Y= 64
X= 5        Y= 125
 1  1
 1  2
 2  1
 2  2
 3  1
 3  2
Ok
```

No se cruzan los ciclos

OCT $

OCT$(expresión entera)

Devuelve una constante alfanumérica que representa el valor octal de **expresión entera**

expresión entera es una expresión numérica en el rango de **-32768 a 65535.**

Si **expresión entera** es negativa, se utiliza la fórmula de complemento a 2. Esto es, **OCT$ (-n)** es igual a **OCT$(65536-n).**

Ejemplo:

```
10 PRINT " X    OCT(X)"
20 FOR J=1 TO 15 STEP 2
30 PRINT USING "##    &&";J;OCT$(J)
40 NEXT J
50 END
RUN
 X    OCT(X)
 1    1
 3    3
 5    5
 7    7
 9    11
11    13
13    15
15    17
Ok
```

ON ERROR GOTO

ON ERROR GOTO número de línea

Para habilitar la interrupción para detección de errores y especificar la primer línea de la subrutina de manejo de los mismos.

Una subrutina de manejo de errores contiene las funciones **ERR** o **ERL** para el procesamiento de errores, y finaliza con la instrucción **RESUME.**

Una vez que se ha habilitado la detección, serán captados todos los errores, inclusive los errores en modo directo (por ej.: SN(Syntax error) (error de sintaxis), provocará un salto a la subrutina mencionada).

Sin embargo, si ocurre un error dentro de la subrutina de manejo de errores, se ignora la interrupción y el sistema obra de la forma acostumbrada (imprime mensaje de error y vuelve al modo comando).

Si no existe **número de línea**, resultará un error con el mensaje **"Undefined line number"** (número de línea inexistente).

Para desactivar la detección de errores se deberá ejecutar un **ON ERROR GOTO Ø** Los siguientes errores provocarán la impresión de un mensaje de error y la detención de la ejecución del programa.

Se recomienda que todas las subrutinas para detección de errores ejecuten un **ON ERROR GOTO Ø** , por si se encuentra un error para el cual no haya posibilidad de una acción recuperatoria.

Asimismo, al finalizar el programa ejecute la sentencia ON ERROR GOTO Ø, caso contrario, quedará activa la rutina de manejo de errores aún en el modo comando.

Ejemplo:

```
10 ON ERROR GOTO 60
20 CLS: PRINT "CALCULO DE RAICES CUADRADAS"
30 INPUT "INGRESE UN NUMERO (O=fin)";N:IF N=O
THEN 70
40 R=SQR(N):PRINT "√DE";N;"=";R
50 GOTO 30
60 IF ERR=5 THEN PRINT "RAICES CUADRADAS DE
NUMEROS NEGATIVOS NO SE PUEDEN CALCULAR!":
RESUME 30
70 ON ERROR GOTO O
80 END
RUN
CALCULO DE RAICES CUADRADAS
INGRESE UN NUMERO (O=fin)? 5
√DE 5 = 2.2360679774998
INGRESE UN NUMERO (O=fin)?-3
RAICES CUADRADAS DE NUMEROS NEGATIVOS NO SE
PUEDEN CALCULAR!
INGRESE UN NUMERO (O=fin)? O
Ok
```

(Vemos que al activar la rutina de detección de errores, en vez de imprimir el mensaje **"Illegal function call"**

(llamado a función incorrecta) hacemos bifurcar el programa a una rutina creada por nosotros que imprime nuestro mensaje de error. Al finalizar el programa hacemos ON ERROR GOTO Ø para desactivar dicha detección.)

Nota: para obtener el símbolo √ pulse las teclas [GRAPH] + [7].

ON GOSUB/ON GOTO

ON expresión entera GOSUB número de línea [, número de línea ...]

ON expresión entera GOTO número de línea [, número de línea ...]

Para derivar a uno de los varios **número de línea** especificados, dependiendo del valor devuelto cuando la **expresión entera** sea resuelta.

El valor de **expresión entera** determina que **número de línea** de la lista será utilizado para la bifurcación, siendo el valor 1 para el primer **número de línea**, 2 para el segundo, etc. Si dicho valor no es un entero, se descartará la fracción.

En la declaración ON GOSUB, cada **número de línea** en la lista debe ser el primer número de línea de una subrutina.

Si el valor de **expresión entera** es cero o mayor que el número de items en la lista o se omitió el **número de línea** en la posición especifica el programa continuará con la próxima instrucción ejecutable.
Pero si **expresión entera** es negativa o mayor que 255, se producirá un error por **'illegal function call'** (llamado a función incorrecto).

Ejemplo:

```
10 INPUT "SELECCIONE NO.(1-3)";A
20 ON A GOTO 40,50,60
30 GOTO 10
40 PRINT "A=1":GOTO 10
50 PRINT "A=2":GOTO 10
60 PRINT "A=3"
70 END
RUN
```

```
SELECCIONE NO.(1-3)? 2
A=2
SELECCIONE NO.(1-3)? 1
A=1
SELECCIONE NO.(1-3)? 6
SELECCIONE NO.(1-3)? 3
A=3
Ok
```

ON INTERVAL GOSUB

ON INTERVAL = intervalo de tiempo GOSUB número de línea

Se utiliza para definir un **intervalo de tiempo** luego del cual se produce la interrupción del programa y se ejecuta la subrutina de interrupciones que comienza en **número de línea.**

Genera una interrupción de tiempo a cada **intervalo de tiempo / 5Ø** de segundo (p.ej. si tomamos **intervalo de tiempo=1** entonces se considera **1/5Ø** de segundo).

Cuando ocurre la detección se ejecuta automáticamente un **INTERVAL STOP** de manera que nunca puedan tener lugar nuevas detecciones que se presenten.

El **RETURN** desde la rutina de detección automáticamente producirá un **INTERVAL ON** a menos que se haya ordenado explícitamente un **INTERVAL OFF** dentro de la misma.

La detección **NO** tiene lugar cuando el **BASIC** no está ejecutando un programa o cuando ejecuta una sentencia **INPUT.**

Cuando una detección de error (resultado de una instrucción **ON ERROR**) tiene lugar, se inhabilitan automáticamente todas las demás captaciones (incluyendo **ERROR, STRIG, STOP, SPRITE, INTERVAL** y **KEY**).

Ver ejemplo en la instrucción INTERVAL ON / OFF.

ON KEY GOSUB

ON KEY GOSUB **número de línea**[,**número de línea...**]
ON KEY GOSUB [[**número de línea**],...] **número de línea**

Para establecer una bifurcación al **número de línea** cuando se pulsa una tecla de función.

Los **número de línea** se colocan en la sentencia en el mismo orden que el de la tecla a la que se refieren, con lo cual a cada tecla le corresponde su **número de línea.** Si se omite algún **número de línea,** la correspondiente tecla de función no causará una interrupción si se pulsa.

Cuando ocurre una detección, se ejecuta automáticamente un **KEY(n) STOP**, de manera tal que nunca se pueda confundir esta detección con una nueva.

El **RETURN** desde la rutina de detección automáticamente hará un **KEY(n)**, a menos que un **KEY(n) OFF** se halla ejecutado dentro de la rutina de detección.

Cuando **BASIC** no se halla ejecutando un programa, no tiene lugar esta detección.

Cuando tiene lugar una detección de error (resultado de una declaración **ON ERROR**), ésta automáticamente desactiva toda otra detección (inclusive **ERROR, STRIG, STOP, SPRITE, INTERVAL** y **KEY**).

Ejemplo:

```
10 ON KEY GOSUB 50,60,70,80
20 KEY(1)ON: KEY(2)ON
30 KEY(3)ON: KEY(4)ON
40 GOTO 40
50 N=1:GOTO 90
60 N=2:GOTO 90
70 N=3:GOTO 90
80 N=4
90 PRINT USING "Ha pulsado la tecla F-##";N
100 RETURN
RUN
Ha pulsado la tecla F- 1
Ha pulsado la tecla F- 3
Ha pulsado la tecla F- 2
Ha pulsado la tecla F- 4
```

(Pulse [CTRL] + [STOP] para finalizar).

ON SPRITE GOSUB

ON SPRITE GOSUB número de línea

Para establecer una bifurcación al **número de línea** cuando se detecte la coincidencia de los **sprites** en cualquier parte de un programa.

Cuando ocurra dicha coincidencia, o sea cuando dos sprites se superponen en la pantalla, automáticamente se ejecuta un **SPRITE STOP**, de manera tal que nunca se pueda tomar como una nueva detección a la anterior.

El **RETURN** desde la rutina de detección hará un **SPRITE ON** a menos que un **SPRITE OFF** haya sido realizado dentro de la rutina de detección.

Las detecciones no tiene lugar cuando el **BASIC** no se halla ejecutando un programa ni cuando se ejecuta una sentencia **INPUT.**

Cuando tiene lugar una detección de error (resultante de una instrucción **ON ERROR**), ésta desactiva automáticamente toda captación (incluyendo **ERROR, STRIG, STOP, SPRITE, INTERVAL** y **KEY**).

Ejemplo:

```
10 A$="-~<<<<C<"
20 B$="<C~<<<~-"
30 SCREEN 2
40 ON SPRITE GOSUB 140
50 SPRITE$(0)=A$
60 SPRITE$(1)=B$
70 SPRITE ON
80 XA=RND(1)*100
90 XB=RND(1)*100
100 FOR Y=0 TO 191
110 PUT SPRITE 0,(50+XA,Y),6
120 PUT SPRITE 1,(50+XB,191-Y),3
130 NEXT Y:GOTO 70
140 SPRITE OFF
150 PLAY"L4CEDFDECREFGAGFER"
```

```
160 IF PLAY(0) THEN 160
170 PUT SPRITE0,(0,208)
180 PUT SPRITE1,(0,208)
190 Y=191:RETURN
RUN
```

(Verá circular por la pantalla dos personajes, que cuando choquen se detendrán y se toca una tonada. Pulse CTRL + STOP para finalizar)

ON STOP GOSUB

ON STOP GOSUB número de línea

Para establecer una bifurcación al **número de línea** cuando se opriman las tecla CTRL + STOP.

Cuando ocurre esta detección, automáticamente se ejecuta un **STOP OFF**, de manera tal que nunca pueda ser confundida la misma con una nueva.

El **RETURN** desde la rutina de detección provocará automáticamente un **STOP ON** a menos que un **STOP OFF** haya sido realizado dentro de la rutina de detección.

La detección no tiene lugar cuando el **BASIC** no se halla ejecutando un programa.

Cuando tiene lugar una detección de error (resultante de una declaración **ON ERROR**), ésta desactiva automáticamente toda detección (incluyendo **ERROR, STRIG, STOP, SPRITE, INTERVAL** y **KEY**).

El usuario debe tener mucho cuidado cuando utilize esta declaración, dado que el programa no podrá detenerse, a menos que ejecute un "reset" del sistema (apagar la máquina), o que en la rutina se prevea un modo de detener el programa como en el ejemplo.

Ejemplo:

```
10 ON STOP GOSUB 70
20 STOP ON
30 INPUT A$
40 PRINT A$
50 IF A$="FIN" THEN STOP OFF:END
60 GOTO 30
70 PRINT "POR FAVOR DIGITE: 'FIN'"
80 RETURN 30
RUN
?TALENT-MSX
TALENT-MSX
? [CTRL] + [STOP]
POR FAVOR DIGITE: 'FIN'
?FIN
FIN
Ok
```

ON STRIG GOSUB

ON STRIG GOSUB número de línea [,número de línea...]
ON STRIG GOSUB [[número de línea],...]número de línea

Para establecer una bifurcación al **número de línea** cuando se pulsa la barra espaciadora o cualquier tecla de disparo de un Joystick .

Se pueden especificar hasta cinco **número de línea.** Los números de teclas de disparo varían de Ø a 4 y corresponden a cada **número de línea** especificado.

Ø Barra espaciadora en el teclado.
1 Primer disparador del joystick conectado en la entrada 1.
2 Primer disparador del joystick conectado en la entrada 2.
3 Segundo disparador del joystick conectado en la entrada 1.
4 Segundo disparador del joystick conectado en la entrada 2.

Los **número de línea** se separan con comas **(,).**

Cuando ocurre una detección, se ejecuta automáticamente un **STRIG(n) STOP,** de manera tal que nunca puedan tener lugar nuevas captaciones que se presenten.

El **RETURN** desde la rutina de detección automáticamente hará un

STRIG(n) ON, a menos que un **STRIG(n) OFF** se halla ejecutado dentro de la rutina de detección.

Cuando **BASIC** no se halla ejecutando un programa, no tiene lugar una detección forzada.

Cuando tiene lugar una detección de error (resultado de una declaración **ON ERROR**), ésta automáticamente desactiva toda captación (inclusive **ERROR, STRIG, STOP, SPRITE, INTERVAL** y **KEY**).

Ejemplo:

```
10 CLS
20 ON STRIG GOSUB 80
30 STRIG(0) ON
40 X=X+.1:PRINT USING "X=##.#   LOG(X)=##.
####";X,LOG(X)
50 GOTO 40
60 '
70 '
80 PRINT:PRINT:PRINT "UNA PAUSA...Y SEGUIMOS
(PULSE RETURN)"
90 IF INKEY$<>CHR$(13) THEN 90 ELSE RETURN
RUN
```

(Aparece la tabla de logaritmos naturales desde Ø en adelante. Si Ud. pulsa barra espaciadora aparece)

```
UNA PAUSA...Y SEGUIMOS (PULSE RETURN)
```

[RETURN]

(Continúa la tabla. Detener el programa con [CTRL] + [STOP] ya que las líneas 4Ø-5Ø conforman un ciclo cerrado).

OPEN

OPEN " dispositivo: [nombre del archivo] " [FOR modo] AS [#] número de archivo

Establece la existencia de un archivo en **dispositivo**, quedando etiquetado el mismo con su **número de archivo.** Además en esta instrucción se describe las características del mismo por medio del **modo.**

Los códigos válidos para **dispositivo** son los siguientes:

CAS:	cassette
CRT:	pantalla
GRP:	pantalla modo alta resolución
LPT:	impresora

El **dispositivo** GRP se utiliza para poder imprimir caracteres alfanuméricos en modo mediana y alta resolución (SCREEN 2 y 3), ya que no está disponible la sentencia **PRINT.**

Se pueden agregar más códigos válidos para **dispositivo** utilizando un cartucho **ROM** o un periférico, por ejemplo un drive de diskettes. Ver el manual del dispositivo correspondiente para su código.

modo puede ser uno de los siguientes:

OUTPUT:	escribir en modo secuencial
INPUT:	leer en modo secuencial
APPEND:	agregar en modo secuencial

número de archivo es una expresión entera cuyo valor está entre 1 y el máximo número de archivos especificado en una declaración **MAXFILES.**

El ingreso de datos se efectúa por intermedio de la sentencia **INPUT #número de archivo** y el egreso de datos, por medio de **PRINT #número de archivo**
Ambas sentencias pueden utilizarse mientras el archivo permanezca abierto (no se ejecute la sentencia **CLOSE** para ese **número de archivo).**

Antes de ejecutar las siguientes sentencias:

PRINT #, PRINT # USING
INPUT #, LINE INPUT #
INPUT$, GET, PUT

el correspondiente **número de archivo** debería estar abierto (**OPEN**). Ver las correspondientes instrucciones para el uso de las sentencias nombradas. Para **GET** y **PUT**, ver el manual del sistema de diskettes

Ejemplo:

```
10 OPEN "CAS:DATA" FOR INPUT AS #1
20 LINE INPUT#1,A$
30 PRINT A$
40 CLOSE
50 END
RUN
```

(Se lee un archivo llamado **DATA** desde cassette y se ingresa la primer línea almacenada en éste. Previamente debió grabarse el dato en dicho archivo.)

OUT

OUT dirección de puerta, expresión entera

Para enviar datos de un byte a periféricos internos o externos de l máquina a través de la puerta de entrada/salida (**I/O port**)

Por periféricos internos entendemos al **PSG** (generador de sonido), e **PPI** (interfase programable para periféricos), y el **VDP** (procesador d pantalla).

dirección de puerta es una expresión entera que se halla en el rang de **Ø a 255**, al igual que **expresión entera.**

expresión entera es el dato (byte) a transmitirse.

Para detalles de **dirección de puerta**, vea el Apéndice E.

Si se envía un dato erróneo con la instrucción puede producirse u bloqueo del sistema que puede detenerse únicamente apagando computadora.

Ejemplo:

```
10 OUT &HA0,7:OUT &HA1,7
20 OUT &HA0,8:OUT &HA1,8
30 FOR I= 0 TO 31
40 OUT &HA0,6:OUT &HA1,I
50 FOR J= 1 TO 50:NEXT J
60 NEXT I
70 PRINT "pulse las teclas <CTRL>+<STOP>"
```

```
80 END
RUN
pulse las teclas <CTRL>+<STOP>
Ok
```

(Se activa el **PSG** y se escucha un ruido hasta que se pulsa lo indicado).

PAD

PAD(expresión entera)

Retorna los distintos estados del dispositivo **"touch pad"** y donde **expresión entera** puede estar en el rango de **Ø a 7.**

Cuando **expresión entera** está en el rango de **Ø a 3,** selecciona el **"touch pad"** conectado a la puerta **1** del joystick, en cambio cuando está en el rango de **4 a 7** selecciona el de la puerta **2.**

Para cada uno de los valores que tome **expresión entera,** serán distintos los valores que se obtienen, y responden al siguiente detalle:

Ø ó 4 : devuelve **-1** cuando se toca el **"touch pad"** y **Ø** cuando no.

1 ó 5 : devuelve la coordenada **X** (horizontal) del punto presionado.

2 ó 6 : devuelve la coordenada **Y** (vertical) del punto presionado.

3 ó 7 : devuelve **-1** cuando se toca el pulsador del **"touch pad"** y **Ø** cuando no.

Cuando **PAD(Ø)** es evaluado, son afectados **PAD(5)** y **PAD(6)** y en cambio cuando **PAD(4)** es evaluado, son afectados **PAD(1)** y **PAD(2).**

Ejemplo:

```
10 SCREEN 2
20 IF PAD(0)=0 THEN SW=0
30 X=PAD(1)
40 Y=PAD(2)
50 IF SW=0 THEN PSET(X,Y) ELSE LINE-(X,Y)
60 SW=1
70 GOTO 20
RUN
```

(Moviendo el **touchpad,** podrá ir dibujando en la pantalla líneas y puntos)

PAINT

PAINT (X,Y) ó STEP(X,Y) [, color de superficie [, color de borde]]

Para pintar una figura cerrada por líneas o por el borde de la pantalla, a partir del punto especificado de coordenadas (X,Y). Este no debe pertenecer a ningún punto del borde, ni, obviamente, al exterior de la figura. Caso contrario, no pinta nada o se pinta toda la pantalla.

Dichas coordenadas son absolutas, o sea, están referidas al punto (Ø,Ø) de la pantalla que es el borde superior izquierdo. Si se especifica STEP(X,Y), las coordenadas quedan referidas al último punto impreso.

Para más detalles de las coordenadas (X,Y), ver la descripción en la instrucción PUT SPRITE.

La instrucción PAINT no admite que (X,Y) quede fuera de la pantalla.

PAINT no debe tener color de borde para gráficos de alta resolución (SCREEN 2) y los bordes pueden ser especificados solamente en la modalidad multicolor o sea SCREEN 3.

En el modo de gráficos de alta resolución color de superficie es considerada como el color del borde. Si no se coloca color de superficie se supone que es el color de frente especificado en la sentencia COLOR.

Ejemplo:

```
 10 SCREEN 2:COLOR 15,1,1:CLS
20 LINE(0,88)-(255,88),14
30 LINE(129,88)-(18,192),14
40 LINE(129,88)-(200,192),14
50 PAINT (110,110),14
60 LINE(0,89)-(255,89),12
70 LINE(121,89)-(5,192),12
80 LINE(137,89)-(215,192),12
90 PAINT (10,120),12:PAINT(210,120),12
100 LINE(0,87)-(255,87),7
110 FOR I=1 TO 70:PSET(INT(RND(1)*255),
INT(RND(1)*80)),15:NEXT
120 OPEN"GRP:" FOR OUTPUT AS#1
130 FOR I=0 TO 80 STEP 2
```

```
140 C=C+1:IF C<3 OR C>15 THEN C=3
150 PSET(I*1.5+50,(9-I/4)^2),1:COLOR
C:PRINT#1,"TALENT MSX"
160 NEXT I:COLOR 15,4,4
170 GOTO 170
RUN
```

(Aparece un gráfico interesante... Pulse CTRL + STOP para finalizar).

PDL

PDL (expresión entera)

Devuelve el valor de estado de un dispositivo **"paddle"** cuyo número especifica **expresión entera.**

Hasta 12 **"paddles"** se pueden conectar en las puertas de joystick de su computadora. El número de puerta seleccionada se especifica con el valor de **expresión entera:**

Cuando el valor es 1, 3, 5, 7, 9 ú 11: conector **Joystick 1.**
Cuando el valor es 2, 4, 6, 8, 1Ø o 12: conector **Joystick 2.**

El valor regresado por **PDL** es un entero de **Ø a 255.**

Ejemplo:

PDL (2)
(Devuelve un valor de acuerdo a la acción ejecutada con el **"paddle"** conectado a la puerta 2 del joystick)

PEEK

PEEK (posición de memoria)

Devuelve el byte (entero decimal en el rango de **Ø a 255**) leído en **posición de memoria.**

posición de memoria debe estar en el rango entre **&HØ** y **&HFFFF** (**Ø** y **65535**). También es válida la representación decimal con signo: desde **-32768** hasta **65535.**

PEEK es la función inversa de la instrucción **POKE**.

Ejemplo:

```
10 CLS
20 FOR I=&H3D75 TO &H4075
30 A=PEEK(I)
40 IF A>32 THEN PRINT CHR$(A);:ELSE PRINT " ";
50 NEXT I
60 GOTO 60
RUN
```

(Aparecen en pantalla los mensajes de error almacenados en la memoria de la máquina que vienen de fábrica. Finalizar con [CTRL] + [STOP]).

PLAY

PLAY expresión alfanumérica para canal A [, expresión alfanumérica para canal B[, expresión alfanumérica para canal C]]

Para ejecutar música de acuerdo al macro-lenguaje de música de **MSX**.

PLAY implementa un concepto similar al del **DRAW** mediante un **"macro-lenguaje musical"**.

Dentro de la cadena de caracteres, **expresión alfanumérica para canal n** consiste en comandos musicales representados por caracteres individuales.

Cuando se especifica un alfanumérico nulo ("") el canal correspondiente permanece en silencio.

Los comandos de caracteres simples en **PLAY** son:

A hasta G con **#**, + ó -, opcionales
(;) toca la nota indicada en la octava presente
(#) ó (+) a continuación, indican un sostenido
(-) indica un bemol.

Esta notación corresponde al cifrado americano (para los que saben música).

On : Establece la octava para las notas siguientes a la instrucción.
Hay 8 octavas, numeradas de **1** a **8** y cada una va desde **C** hasta **B**.
La Octava **4** es el valor supuesto si no se coloca **On**.

Nn : Ejecuta la nota **n** que puede variar desde **Ø** a **96**, y n = **Ø** significa silencio.
Esta es una forma alternativa para seleccionar las notas, además de la que especifica la octava (**On**) y la designación de la nota (**A-G**). (El **do** (C) de la octava 4 es 36).

Ln : Establece la duración de las notas siguientes a la instrucción. La duración establecida de la nota es **1/n**. **n** puede tomar valores de **1** a **64**. La siguiente tabla quizás sea más ilustrativa:

L1 nota entera (redonda)
L2 media nota (blanca)
L3 una de un triplete de 3 medias notas (tresillo)
L4 un cuarto de nota (negra)
L5 uno de un quinteto (quintillo)
:

La duración también puede ir a continuación de la nota cuando Ud. desee cambiar sólo la duración de la misma.
Por ejemplo, **A16** es equivalente a **L16A**. **4** es el valor supuesto si no se coloca **Ln**.

Rn : indica **Pausa** (silencio) . **n** puede variar desde **1** a **64**, y determina la longitud de la misma al igual que en **L** (largo) **4** es el valor supuesto si no se coloca **n**.

(.) : Después de una nota, provoca que la nota sea ejecutada como una con **puntillo**. Esto significa que su longitud sea multiplicada por 3/2. Puede aparecer más de un punto después de la nota y la duración será ajustada de acuerdo a éstos.
Por ejemplo: "A..." tendrá un largo de 27/8.
Los puntos pueden aparecer también después de una Pausa (**R**) para establecer su largo de la misma forma.

Tn : **Tempo**. Establece el número de cuartos de nota en un minuto. **n** puede variar desde **32** a **255** y **12Ø** es el valor supuesto si no se coloca **Tn**.

Vn : **Volumen.** Regula el volumen de la salida de audio. **n** puede variar de **Ø** a **15** . **8** es el valor supuesto si no se coloca **Vn.**

Mn : **Modulación.** Establece los períodos de envolvente. **n** puede variar desde **1** a **65535. 255** es el valor supuesto si no se coloca **Mn.**

Sn : **Forma.** Establece la forma de la envolvente. **n** puede variar de **1** a **15. 1** es el valor supuesto si no se coloca **Sn.**

En todos estos comandos el argumento **n** puede ser una constante como **12** o una variable.
En este caso, deberá tener el formato : **"Comando=variable numérica;"**; donde **variable numérica** es el nombre de la variable de que se trata.

El punto y coma (;) se requiere cuando se utiliza un comando de esta forma.

Nota: todos los valores especificados con los comandos mencionados serán reestablecidos a los valores de norma cuando se genere un sonido **BEEP.**

Ejemplo:

```
10 CLS:PRINT"CANON":BEEP
20 A$="T80L8CDEC16R16CDECEFG4L8EFG8R8G1604A1604
L16GFL8ECD03G04C4L8D03G04C2"
30 B$="V6T80R4L804CDCCCDECF4L8EFL4DFCECE03G2"
40 C$="V6T80R403L4GDGDL8BDL4BGBGDCG04E2"
50 PLAY A$,B$,C$
RUN
```

(Se escucha un pequeño canon)

PLAY < n >

PLAY (expresión entera)

Devuelve el estado de la ejecución musical activada por **PLAY.**

expresión entera puede estar en el rango de **Ø** a **3.**

Si **expresión entera = Ø** se le aplica al estado de los tres canales la operación lógica **OR** y ese es el valor obtenido.

expresión entera= 1, 2 o 3, devuelve el valor -1 si aún se está ejecutando la música en el canal correspondiente. Caso contrario devuelve un Ø.

Inmediatamente después de ejecutarse la instrucción **PLAY**, esta función devuelve -1, sin importar el estado de la ejecución en ese momento.

Ejemplo:

```
10 'BACH
20 PLAY"T200L6V12","T200L2V9"
30 PLAY"R8O6GAB07DCCED","O4G05GE"
40 PLAY"DGF#GDO6BGAB","O4B05EO4E"
50 PLAY"O7CDEDCO6BGABG","O4AB05C"
60 PLAY"F#GADF#AO7CO6BA","O5DF#D"
70 PLAY"BGAB07DCCED","GGC"
80 PLAY"DGF#GDO6BGAB","O4B05ED"
90 PLAY"EO7DCO6BAGDGF#G2","CC#DG"
100 IF PLAY(0) THEN 100
110 PRINT " *** FIN ***"
120 END
RUN
(Se oye una melodía conocida de Bach).
 *** FIN ***
Ok
```

POINT

POINT (coordenada X,coordenada Y)
POINT STEP(coordenada X,coordenada Y)

Devuelve el código de color del punto de coordenadas **(X,Y)** especificado en modo gráfico **(SCREEN 2** y **SCREEN 3).** Si se especifica **STEP(X,Y)** se toma las coordenadas relativas al último punto referenciado como origen.

El último punto referenciado no se altera luego de ejecutar la sentencia **POINT.**

Para más detalles de las coordenadas **(X,Y)** ver la instrucción **PUT SPRITE.**

Ejemplo:

```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS#1
30 FOR H=1 TO 20
40 FOR J=1 TO 8
50 PRESET(J*24,H*8)
60 C=INT(RND(1)*13)+1
70 COLOR C:PRINT #1,"♥"
80 NEXT J,H
90 COLOR 15
100 PRESET(100,170)
110 PRINT #1,"COLOR No."
120 FOR H=1 TO 20
130 FOR J=1 TO 8
140 K=POINT(J*24+4,H*8+4)
150 PRESET(J*24+8,H*8)
160 PRINT #1,USING"##";K
170 NEXT J,H
180 GOTO 180
RUN
```

♥ 9	♥ 3	♥11	♥ 9	♥11	4	♥ 6	♥14	♥10
♥ 8	♥12	♥ 8	4	♥ 3	♥ 8	♥12	♥ 2	♥ 8
♥10	♥ 7	♥12	♥12	♥ 9	♥ 2	♥ 5	♥ 6	4
4	♥12	♥ 3	♥ 5	♥ 7	♥ 7	♥ 8	♥14	♥10
♥ 3	♥ 3	♥ 2	♥ 3	♥ 3	♥ 6	12	♥ 9	♥14
♥ 6	4	♥14	♥ 8	♥ 3	♥ 8	♥10	♥14	4
♥13	♥10	♥11	4	♥11	♥12	♥ 4	♥ 5	♥ 9
♥ 5	♥ 7	♥10	♥ 9	♥ 5	♥14	♥12	♥ 3	♥12
♥ 5	♥14	♥ 2	♥ 3	♥12	♥ 2	♥ 8	4	♥ 9
♥11	♥ 3	♥ 8	♥14	4	♥14	6	♥13	♥ 8
4	♥ 9	♥11	♥ 7	♥ 8	♥10	♥13	♥14	♥ 6
♥ 9	♥12	♥10	♥13	♥ 2	♥ 3	♥ 4	♥ 2	♥ 6
♥11	♥ 9	♥ 2	♥ 8	♥ 9	♥ 7	♥12	♥12	♥ 2
♥ 5	♥13	♥11	♥12	♥13	♥ 9	♥ 7	♥ 3	♥ 7
♥12	♥14	♥11	♥ 7	♥ 6	♥13	♥ 3	♥11	♥14
♥ 3	♥ 9	♥11	♥ 5	♥ 9	♥ 6	♥ 7	♥12	♥ 5
4	♥10	♥ 5	♥12	♥ 2	♥ 8	♥13	♥ 5	4
4	4	♥ 6	♥12	♥ 9	♥ 7	♥11	♥11	♥ 5
♥ 2	4	♥14	♥ 2	♥12	♥ 8	♥ 7	♥10	♥ 2
♥ 2	♥10	♥ 9	♥ 6	♥ 6	♥11	11	♥ 5	♥ 9
♥ 9	♥14	♥13	♥ 2	♥ 9	♥11	10	♥11	♥ 5

Color No.

POKE

POKE posición de memoria, expresión entera

Envía un byte (número entero en el rango de **Ø a 255**) de acuerdo a **expresión entera** , a la **posición de memoria** especificada.

posición de memoria debe estar en el rango entre **&HØ** y **&HFFFF.**

POKE es la función inversa de la instrucción **PEEK.**

Para ver detalles del mapa de memoria, ver el Apéndice **D.**

Si se ejecuta erróneamente esta instrucción es posible que la computadora se "bloquee", y la única forma de recuperar el control es apagarla y volverla a encender.

Ejemplo:

Ver el ejemplo en la instrucción USR.

POS

POS (expresión)

Devuelve la posición en la que se encuentra el cursor en una pantalla modo texto, donde la posición más a la izquierda es **Ø.**

expresión es un argumento ficticio.

Ejemplo:

```
10 CLS
20 FOR J=0 TO 9
30 LOCATE J*2,J
40 PRINT POS(X)
50 NEXT J
60 END
```

```
RUN
 0
    2
       4
          6
             8
                10
                   12
                      14
                         16
                            18
Ok
```

PRESET

PRESET (X,Y) [, código de color]
PRESET STEP(X,Y) [, código de color]

Para reemplazar el color del punto cuya coordenada es especificada por **(X,Y)** con el color indicado en **código de color.**

Cuando se omite **código de color**, el punto representado tendrá el color-fondo de la pantalla indicado en la sentencia **COLOR.**

Si las coordenadas **(X,Y)** se especifican en formato relativo **STEP(X,Y)**, se toma como origen el último punto referenciado.

Luego de ejecutarse la instrucción **PRESET**, el último punto referenciado pasa a ser el indicado en **(X,Y).**

Para más detalles del índice de coordenadas **(X,Y)** ver su descripción en la instrucción **PUT SPRITE.**

Ejemplo:

```
10 SCREEN 2
20 LINE(10,10)-(245,180),15,BF
30 FOR I=0 TO 700
40 X=INT(RND(1)*233)+11
50 Y=INT(RND(1)*168)+11
60 PRESET(X,Y)
```

```
70 NEXT I
80 GOTO 80
RUN
```

(Se dibuja un rectángulo blanco y luego se van apagando puntos al azar. Pulse CTRL + STOP para finalizar)

PRINT

PRINT [expresión] [separador campo] [expresión] [separador campo]...

Para dar salida por pantalla en modo texto a expresiones o variables numéricas o alfanuméricas.

expresión: es el detalle de la información a imprimir, pero si son omitidos se imprimirá una línea en blanco.

DATOS NUMERICOS: se imprime el contenido de la expresión o la variable especificada con la puntuación decimal correspondiente.

Los números positivos siempre van precedidos por un espacio en blanco y los negativos por un signo menos.

Ejemplo:

```
10 A=123
20 PRINT "PRUEBA";A;A
30 PRINT "PRUEBA";A*-1;A*-1
40 END
RUN
PRUEBA 123  123
PRUEBA-123 -123
Ok
```

Cuando una variable numérica de simple o doble precisión excede los 14 dígitos significativos, ésta se imprime con notación exponencial o científica (base 10).

Ejemplo:

```
10 CLS
20 A!=999999!
30 A#=9999999#
40 FOR J=1 TO 6
50 A!=A!*100
```

```
60 A#=A#*100
70 PRINT A!;A#
80 NEXT J
90 END
RUN
 99999900  999999900
 9999990000  99999990000
 999999000000  9999999000000
 99999900000000  9.999999E+14
 9.99999E+15  9.999999E+16
 9.99999E+17  9.999999E+18
Ok
```

TABULACION: se indica con **TAB(n)** y es utilizada para indicar a partir de cual columna de la línea comenzará el próximo **expresión.** Para más detalles, ver la instrucción **TAB.**

DATOS ALFANUMERICOS: se imprime el contenido de la variable o el de una expresión alfanumérica escrita entre comillas.

Si se desea imprimir un espacio en blanco antes o después, debe incluírselo a la variable o preveer insertarlo como constante entre comillas dentro de la instrucción **PRINT.**

Ejemplo:

```
10 CLS
20 A$="TALENT"
30 B$="MSX"
40 PRINT A$;" ";B$;" EL COMPUTADOR REAL"
50 END
RUN
TALENT MSX EL COMPUTADOR REAL
Ok
```

La posición de cada ítem impreso es determinada por los signos de puntuación que se utilicen para separarlos en la lista. El **BASIC** divide la línea en zonas de impresión de 14 espacios cada uno.

Si se utiliza como **separador campo** una coma, ésta provocará que el próximo valor sea impreso al comienzo de la zona más inmediata; en cambio un punto y coma causará que el próximo valor sea impreso inmediatamente después del último.

Si la instrucción **PRINT** termina **con** una coma o un punto y coma, la próxima instrucción **PRINT** comenzará imprimiendo en la misma línea espaciando, de acuerdo a ésto.

En cambio si el **PRINT** finaliza **sin** una coma o un punto y coma, al final de la línea se imprimirá un retorno de carro (**RETURN**, caracter ASCII 13).

Cuando la línea impresa sea más larga que el ancho de la pantalla, el **BASIC** va a la próxima línea física y continúa la impresión.

En una declaración, en lugar de la palabra **PRINT**, puede ser utilizado un signo de interrogación (**?**).

Ejemplo:

```
10 A=1985
20 A$=" TALENT MSX"
30 PRINT
40 PRINT "COMPUTADORA";A$;" AÑO";A
50 END
RUN

COMPUTADORA TALENT MSX AÑO 1985
Ok
```

(Para obtener la Ñ, pulse [CODE] + [N])

PRINT USING

PRINT USING formato impresión ; expresión [;expresión ...] [;]
PRINT USING formato impresión expresión [;expresión ...] [;]

Para imprimir datos o variables numéricas o alfanuméricas utilizando el **formato impresión** especificado.

formato impresión es una expresión alfanumérica, que contiene caracteres que determinan la longitud del campo, las características y el formateo de la información a imprimir.

En **expresión** se especificarán cada una de las variables numéricas o alfanuméricas que van a ser impresas, separadas por un punto y coma o coma.

IMPRESION DE DATOS NUMERICOS: se deberán emplear alguno de los siguientes caracteres para definir el **formato impresión**:

"#####...#": Para imprimir los **n** dígitos especificados, para cada una de las variables.

Puede insertarse un punto decimal (.) en cualquier posición del **formato impresión**, teniendo en cuenta que si un **"#"** precede al punto decimal, siempre se imprimirá este dígito , aun cuando sea **Ø** si la parte entera es nula.

Si el **formato impresión** es más largo que la **expresión numérica**, la impresión se efectuará justificando a derecha y rellenado con espacios a la izquierda.

Cuando la **expresión numérica** es más larga que el **formato impresión**, si los caracteres que exceden son enteros provocará la impresión del **"%"**, en cambio si los que exceden son decimales, los mismos serán **redondeados** hasta la posicion especificada.

Ejemplo:

```
PRINT USING"###.##";10.2,2,3.456,0.24
 10.20  2.00  3.46  0.24
Ok
```

"+": Un signo **más** al comienzo o al final del **formato impresión** determinará que el signo del número (más o menos) sea impreso antes o después del número.

Ejemplo:

```
PRINT USING"+###.##";1.25;-1.25
  +1.25  -1.25
Ok
PRINT USING"###.##+";1.25;-1.25
  1.25+  1.25-
Ok
```

"-": Un signo **menos** al final del **formato impresión**, hará que únicamente cuando los números sean negativos, el signo menos será impreso tras los mismos.

Ejemplo:

```
PRINT USING"###.##-";1.25;-1.25
  1.25   1.25-
Ok
```

""**: Un **doble asterisco** al principio del **formato impresión** determinará que los digitos no significativos a la izquierda de cada **expresión numérica** sean completadas con asteriscos y especifican posiciones para dos o más dígitos.

Ejemplo:

```
PRINT USING"***.###";1.25;-1.25
**1.250*-1.250
Ok
```

"$$": Un **doble signo pesos** al principio del **formato impresión** determinará que un signo pesos se imprima inmediatamente a la izquierda de cada **expresión numérica**, especificando dos posiciones más, una de las cuales es para el signo pesos.

Los números negativos no se pueden utilizar, a menos que el signo menos se ubique a la derecha, trás el número.

Ejemplo:

```
PRINT USING"$$###.##";12.35;-12.35
  $12.35 -$12.35
Ok
PRINT USING"$$###.##-";12.35;-12.35
  $12.35   $12.35-
Ok
```

"$"**: Un **doble asterisco y un signo pesos** al principio del **formato impresión** determinará que se combinen ambos efectos. Es decir, los espacios a la izquierda serán completados con **asteriscos** y el signo **$** se imprimirá a la izquierda del número. Con esto se especifican tres posiciones más, una de las cuales es el signo pesos.

Ejemplo:

```
PRINT USING"**$#.##";12.35;-12.35
*$12.35-$12.35
Ok
```

",": Una **coma** situada a la izquierda del punto decimal en el **formato impresión** determinará que se imprima una coma cada tercer dígito situado a la izquierda del punto decimal y especifica una posición más.

Cuando la **coma** que se halle al final del **formato impresión** ésta será impresa como parte del mismo, pero no tendrá ningún efecto si se usa en el formato exponencial.

Ejemplo:

```
PRINT USING"####,.##";1234.5
1,234.50
Ok
PRINT USING"####.##,";1234.5
1234.50,
Ok
```

"^^^^": Cuatro símbolos circunflejos pueden colocarse después de los dígitos para especificar **formato exponencial**, reservando espacio para imprimir "E+xx", debiendo especificarse cualquier posición para el punto decimal.

Los dígitos significativos son justificados a la izquierda y el exponente es ajustado. A menos que sea especificado un signo + antepuesto o a continuación, se utilizará una posición a la izquierda del punto decimal para imprimir un espacio o un signo menos.

Ejemplo:

```
PRINT USING"##.##^^^^";234.56
 2.35E+02
Ok
PRINT USING" #.##^^^^";-12.34
 -.12E+02
Ok
PRINT USING"+#.##^^^^";12.34;-12.34
+1.23E+01-1.23E+01
Ok
```

Si la cantidad de dígitos especificados excede a 24, resultará un error con el mensaje **"Illegal function call"**(llamado a función incorrecto).

Ejemplo:

```
10 PRINT USING"######";123456!
20 PRINT USING"#.####";12.345
30 PRINT USING"####.#";1234.56
40 PRINT USING"$$####";12345!
50 PRINT USING"####,#";12345!
60 PRINT USING"*****";12!
70 PRINT USING"#####+";12345!
80 PRINT USING"######^^^^";1234567890#
90 END
RUN
```

```
123456
%12.3450
1234.6
$12345
12,345
***12
12345+
 12346E+05
Ok
```

IMPRESION DE DATOS ALFANUMERICOS: se deberán emplear alguno de los siguientes caracteres para definir el **formato impresión:**

"!": Para imprimir solamente el primer caracter de cada una de las variables alfanuméricas.

Ejemplo:

```
A$="TALENT":PRINT USING"!";A$
T
Ok
```

"! \ **espacios** \": Para imprimir los primeros **(2+n)** caracteres de cada una de las variables.

Si los "\\" son definidos sin espacios intermedios, serán impresos dos caracteres; con un espacio, tres caracteres y así sucesivamente.

Si la **expresión alfanumérica** es más larga que el **formato impresión,** los caracteres que exceden son ignorados.

En cambio, si el **formato impresión** es más largo que la **expresión alfanumérica,** la impresión se efectuará justificando a izquierda y rellenado con espacios a la derecha.

Ejemplo:

```
A$="TALENT":PRINT USING"\   \";A$
TALE
Ok
```

"&": Para imprimir todos los caracteres de cada una de las expresiones alfanuméricas.

Ejemplo:

```
A$="TALENT":PRINT USING"COMPUTADOR & MSX";A$
COMPUTADOR TALENT MSX
Ok
```

En todos los casos se puede reemplazar la instrucción **PRINT** con un signo de pregunta **(?)**

PRINT #/PRINT # USING

PRINT # número de archivo [, expresión [; expresión ....]] [;]
PRINT # número de archivo [, expresión [, expresión ....]] [,]

PRINT # número de archivo USING formato impresión ; expresión [; expresión ....] [;]
PRINT # número de archivo USING formato impresión ; expresión [, expresión ....] [,]

Para escribir uno o más expresiones o variables numéricas o alfanuméricas, a través del canal especificado por **número de archivo.**

El archivo indicado por **número de archivo** debe estar abierto por la sentencia **OPEN** en el modo de salida **(OUTPUT).**

Los formatos de impresión son idénticos a los de las sentencias **PRINT / PRINT USING,** con la diferencia de que en este caso los datos se envían al **número de archivo** especificado.

Se puede especificar más de una **expresión** en una sentencia PRINT # o PRINT # USING, separándolas con comas **(,)** o puntos y comas **(;).**

El formato de salida depende de los separadores utilizados y en el caso de **PRINT# USING,** de **formato impresión.**

Cuando **expresión** es numérica y varias de ellas se separan con comas, se las imprime en los campos, y cada dato puede contener espacios en blanco.

Cuando **expresión** es alfanumérica y varias de ellas se separan con punto y coma **(;),** se imprimen en forma contínua en el archivo. Cuando el mismo se lee con una instrucción **INPUT#** ó **LINE INPUT#,** los datos alfanuméricos ingresan como uno solo.

Si no desea esto último y va a utilizar la instrucción **INPUT#,** separe con comas **(,)** los datos. Asimismo para la instrucción **LINE INPUT#** deberá separar cada dato alfanumérico con un **CR (return,** caracter ASCII **13)** y un **LF (line feed,** caracter ASCII **10)** cuando utiliza la sentencia **PRINT#.**

Si detrás de **expresión** no se especifica nada, **PRINT#** envía la

secuencia return/line feed luego de la impresión. En cambio, si se coloca una coma o un punto y coma no se envía dicha secuencia.

Puede utilizarse el signo de pregunta (?) para reemplazar a **PRINT** (para más detalles ver declaraciones **PRINT/PRINT USING**).

Ejemplo:

```
10 CLS
20 A$="TALENT MSX"
30 OPEN"CRT:"FOR OUTPUT AS #1
40 LOCATE 10,10
50 PRINT#1,A$
60 CLOSE
RUN
TALENT MSX
Ok
```

PSET

PSET (coordenada X , coordenada Y) [, código de color]
PSET STEP (coordenada X , coordenada Y) [, código de color]

Para encender un punto en las coordenadas especificada por **(X,Y)** en una pantalla modo gráfico, con el color especificado por **código de color.**

Si no se especifica **código de color** el punto representado tendrá color-frente de la pantalla especificado en la sentencia **COLOR.**

Si se utilizan coordenadas relativas, o sea, **STEP(X,Y)**, éstas tienen origen en el último punto referenciado.

Cuando se ejecuta la sentencia **PSET**, el último punto referenciado queda actualizado con las coordenadas **(X,Y)**

Para más detalles del Indice de coordenadas **(X,Y)**, ver su descripción en la instrucción **PUT SPRITE.**

Ejemplo:

```
10 CLS:SCREEN 2
20 LINE(49,96)-(208,96):LINE(47,56)-(47,134)
30 FOR I=87 TO208 STEP 40
40 PSET(I,97):PSET(I,98)
50 NEXT I
60 PI=4*ATN(1)
70 FOR RA=0 TO 2*PI STEP .05
80 PF=96-INT(40*SIN(RA)+.5)
90 PC=47+INT(25.46*RA+.5)
100 PSET(PC,PF)
110 PF=96-INT(40*COS(RA)+.5)
120 PSET(PC,PF),3
130 NEXT
140 IF INKEY$="" THEN 140 ELSE END
RUN
```

(Se imprimen las funciones **SIN** y **COS** en dos colores. Pulse cualquier tecla para finalizar).

PUT SPRITE

PUT SPRITE número de plano sprite, (X,Y) [, código de color] [, número de dibujo sprite]
PUT SPRITE número de plano sprite, STEP(X,Y) [, código de color] [, número de dibujo sprite]

Permite dibujar una figura sprite definida en **SPRITE$**, en las coordenadas especificadas en un plano sprite con el número indicado.

número de plano sprite que puede variar de **Ø** a **31.**

Hasta **32** planos sprite están disponibles, correspondiendo a cada una un número de **Ø** a **31.**
Una sola figura sprite puede dibujarse en cada plano sprite, y hasta **32** figuras sprite (planos) pueden dibujarse en pantalla simultáneamente.
Sin embargo, sólo pueden dibujarse **4** sprites en una misma fila.

Los planos sprite de menor **número de plano sprite** tienen mayor prioridad; un sprite de **número de plano sprite** mayor queda oculto detrás del que tienen mayor prioridad cuando se superponen.

La coordenada **X** (posición horizontal) se especifica con una expresión entera cuyo valor está en el rango de **-32** a **255**; la coordenada **Y** (posición vertical) se especifica como otra expresión entera cuyo valor está en el rango de **-32** a **191**.

Cuando las coordenadas se especifican en forma absoluta, **(X,Y)** quedan referidos respecto del borde superior izquierdo de la pantalla cuyas coordenadas son **(Ø,Ø)**.

En cambio, si se especifican coordenadas relativas al último punto referenciado (el último punto que se procesó al momento de la ejecución de **PUT SPRITE**) con **STEP (X,Y)**, éstas coordenadas quedan referidas a dicho punto como origen.

Una vez que se ejecuta **PUT SPRITE**, el último punto referenciado queda actualizado con los valores donde se ubicó la figura sprite.

Ejemplo:

(1Ø, 1Ø) forma absoluta. Está ubicado 1Ø puntos debajo del origen y 1Ø puntos a la derecha del mismo.
STEP (1Ø, Ø) desplazado 1Ø en **X** y Ø en **Y** respecto al último punto referenciado.

Cuando el **BASIC** explora los valores de las coordenadas le permitirá a las mismas ubicarse por encima el borde de la pantalla. No obstante todo valor fuera del rango de los enteros (-32768 a 32767) causará un error de desborde (overflow).

Los valores de coordenadas que caigan fuera de la pantalla serán substituídos, por ejemplo por un **Ø** para cualquier especificación de coordenada negativa.

Siendo **(Ø,Ø)** el borde superior izquierdo, puede parecer extraño el comenzar la numeración de **Y** arriba, de manera que el rincón de abajo a la izquierda es **(Ø,191)**, tanto en modo de mediana resolución como en de alta resolución, pero esta es la norma.

La descripción mencionada anteriormente puede ser aplicada donde quiera se utilicen las coordenadas gráficas.

Para borrar un sprite de la pantalla:

Se le asigna a **Y** el valor: **2Ø8** (**&HDO**), todos los planos por detrás del sprite desaparecerán incluyendo al especificado, hasta que un valor diferente de **2Ø8** sea asignado a dicho plano.

Si especifica el valor **2Ø9** a la coordenada **Y**, sólo desaparece el sprite cuyo número se está indicando.

Cuando se omite **código de color**, este asume el color-frente que se especifica en la sentencia **COLOR.**

número de dibujo especifica el dibujo del sprite que se indica en SPRITE$.

Debe ser menor de **256** cuando el tamaño de los sprites es de **Ø** o **1** y menor que **64** cuando sean de **2** o **3.**
Si se omite **número de dibujo** se considera que **número de dibujo= número de plano sprite** (ver también la declaración **SCREEN** y la variable **SPRITE$**)

Ejemplo:

```
10 DEFINT A-Z
20 DIM X(10),Y(10)
30 SCREEN2,2
40 COLOR ,1,1
50 CLS
60 I=96
70 FOR J=1 TO 10
80 X(J)=I :Y(J)=J*15
90 NEXT J
100 FOR J=0 TO 31
110 READ A$
120 B$=B$+CHR$(VAL("&H"+A$))
130 NEXT J
140 SPRITE $(0)=B$
150 FOR J=1 TO 10
160 PUT SPRITE J,(X(J),Y(J)),J+4,0
170 NEXT J
180 FOR J=1 TO 10
190 X(J)=(X(J)+(RND(1)*21-10))MOD 256
200 Y(J)=(Y(J)+(RND(1)*21-10))MOD 192
210 NEXTJ
220 GOTO 150
230 DATA 0C,06,62,F2,FA,DD,CF,C7
240 DATA FF,7F,3F,1B,37,3E,1C,00
250 DATA 30,30,46,4F,5F,BB,FB,F3,E3
260 DATA FF,FE,FC,D8,EC,7C,38,00
RUN
```

(Aparecen mariposas en la pantalla. Pulse CTRL + STOP para finalizar).

READ

READ variable [,variable...]

Para leer las constantes numéricas y alfanuméricas almacenadas por las líneas de sentencia **DATA** y asignárselo a la(s) **variable(s).**

variable puede ser del tipo numérico ó alfanumérico, separadas por comas.

Las constantes leídas deben coincidir con el tipo de variable especificado en la declaración **READ.** Si así no fuese se emitirá por pantalla el mensaje de error: **"Syntax error"**(error de sintaxis).

Una única instrucción **READ** puede acceder a uno o más datos en orden correlativo, o varios **READ** pueden acceder a un mismo dato.

Si el número de **variable(s)** en la sentencia **READ** excede al número de elementos de la declaración **DATA,** busca la siguiente línea **DATA** y lee los datos de allí. Si no encontrara más datos, resultará el error **"Out of DATA"** (sin datos), pero si el número de variables es menor al número de elementos del **DATA,** los subsiguientes **READ** comenzarán a leer los datos a partir del primer elemento de **DATA** no leído. Si no hubieran subsiguientes comandos **READ,** los datos extra serán ignorados.

Para releer declaraciones **DATA** en una secuencia distinta de la normal, debe utilizarse la instrucción **RESTORE.**

Ejemplo:

```
10 CLS: FOR J=1 TO 3
20 READ A$,B,C
30 PRINT TAB(3);A$;
40 D=B+C
50 PRINT TAB(15) USING"#####";B;C;D
60 NEXT J
70 DATA COMPUTADORA,50,60
80 DATA TALENT,30,40
90 DATA MSX,1,2
RUN
   COMPUTADORA     50    60   110
   TALENT          30    40    70
   MSX              1     2     3
Ok
```

REM

REM [comentarios]
' [comentarios]

Para permitir insertar observaciones explicativas de un programa.

Las declaraciones **REM** no son ejecutadas y se imprimen exactamente como fueron ingresadas al programa cuando éste se **LIST**a.

Un programa puede bifurcar a una sentencia **REM** (desde una declaración **GOTO** o **GOSUB**) y la ejecución continuará con la primer sentencia ejecutable después del **REM**.

Estas observaciones pueden ser agregadas al final de una línea precediéndolas con una sola comilla (') en lugar del **REM**. Si se utiliza una comilla a continuación de una sentencia (en la misma línea) no es necesario colocar el separador de sentencias (:).

No debe utilizarse en un comando **DATA** ya que sería considerado como un dato legal más.

Ejemplo:

```
10 REM Aplicacion de las declaraciones de READ y DATA
20 FOR J=1 TO 3
30 READ A$
40 DATA Computadora,Talent,Msx
50 PRINT A$;SPC(2);
60 NEXT J
70 END
RUN
Computadora  Talent  Msx
Ok
```

RENUM

RENUM [número de línea nuevo] [, número de línea anterior] [, incremento]

Para renumerar líneas de un programa.

número de línea nuevo es el primer número de línea a ser utilizado en la nueva secuencia. Si no se indica, se considera que es **10**

número de línea anterior es la línea del programa en memoria donde comenzará la renumeración. Si no se indica, es la primer línea del programa.

incremento es el incremento que se va a utilizar en la nueva secuencia. Si no se indica, se considera que es **1Ø**

RENUM también cambia automáticamente todos los números de línea de referencia seguidos por los comandos **GOTO , GOSUB, THEN, ELSE, ON..GOTO, ON..GOSUB** y **ERL** determinando el nuevo número.

Si un número de línea inexistente aparece después de uno de estos comandos, se imprime el mensaje de error: **"Undefined line** nnnn **in** mmmm" (Número de línea inexistente **nnnn** en la línea **mmmm**), y el número de referencia incorrecto (**nnnn**) no es cambiado por **RENUM**, pero si puede cambiar el número de línea **mmmm.**

Debe tenerse en cuenta que un **RENUM** no puede utilizarse para cambiar el orden de las líneas de un programa (por ejemplo, **RENUM 15, 3Ø** cuando el programa cuenta con tres líneas numeradas **1Ø, 2Ø, 3Ø** o tampoco crear números de línea mayores de **65529.**
Si así se hiciese aparecería un mensaje de error: **"Illegal function call"** (Llamado incorrecto a función).

Ejemplo:

```
10 REM RENUMERACION
20 '
25 PRINT "HOLA":GOTO 70
55 '
70 END
RUN
HOLA
Ok
RENUM 20,10,5
Ok
list
20 REM RENUMERACION
25 '
30 PRINT "HOLA":GOTO 40
35 '
40 END
Ok
```

RESTORE

RESTORE [número de línea]

Para especificar el **número de línea** de las sentencias de **DATA** desde donde se leerán primeramente los datos con la sentencia **READ.**

Después de que se ejecute una instrucción **RESTORE,** el próximo **READ** accederá al primer ítem de la sentencia **DATA** de menor número de línea del programa.

Si se especifica **número de línea,** el próximo **READ** accederá al primer ítem de los datos de la línea especificada.

En el caso de indicarse un número de línea inexistente resultará el error: **"Undefined line number"** (número de línea inexistente).

Ejemplo:

```
10 RESTORE:READ A,B,C,D,E,F,G
20 CLS:PRINT A,D,F
30 RESTORE 100
40 READ A$,B$,C$,D$,E$,F$,G$
50 PRINT A$;B$;C$;D$;E$;F$;G$
60 END
70 DATA 1,4,6
80 DATA 2,5,3
90 DATA 7,8,9,0
100 DATA D,P,C
110 DATA 2,0,0
120 DATA A,C,D,E,F,G
RUN
 1              2
 3
DPC200A
Ok
```

RESUME

RESUME [Ø]
RESUME NEXT
RESUME número de línea

Para continuar la ejecución de un programa después de realizar un procedimiento de recuperación de un error especificado por la sentencia **ON ERROR GOTO.**

Cualquiera de los tres formatos antes especificados pueden ser utilizados, dependiendo desde donde será **reasumida** la ejecución:

RESUME ó **RESUME Ø** en la instrucción que causó el error.

RESUME NEXT : en la instrucción inmediata siguiente a la que causó el error.

RESUME número de línea: en la instrucción del **número de línea** indicado.

Una declaración **RESUME** que no esté dentro de una subrutina para captura de errores causará un **"RESUME without error"** (**RESUME** sin su error).

Ejemplo:

```
10 ON ERROR GOTO 60
20 INPUT "X=";X
30 PRINT SQR(X);
40 IF F=1 THEN PRINT "i" ELSE PRINT
50 F=0:GOTO 20
60 X=-X:F=1
70 RESUME
RUN
X=? 7
 2.6457513110646
X=?-4
 2 i
X=?
```

(Detener con CTRL + STOP)

RETURN

RETURN [número de línea]

Trabaja en conjunto con las sentencias **GOSUB** y **ON GOSUB** y transfiere el control de vuelta a la instrucción siguiente al **GOSUB** y **ON GOSUB** que la llamó.

Si se coloca el **número de línea**, la transferencia se efectúa al número de línea indicado.

Pueden existir más de un **RETURN** dentro de una línea.

RETURN funciona con las intrucciones: **GOSUB, ON GOSUB, ON INTERVAL GOSUB, ON KEY GOSUB, ON SPRITE GOSUB, ON STOP GOSUB** y **ON STRIG GOSUB.**

Ejemplo:

```
10 CLS: ON STOP GOSUB 100:STOP ON
20 PRINT "TABLAS DE MULTIPLICAR."
30 INPUT "INGRESE UN NUMERO:";A
40 PRINT "TABLA DE MULTIPLICAR DEL";A
50 FOR I=1 TO 10
60 R=I*A
70 GOSUB 130
80 NEXT I
90 GOTO 20
100 INPUT "DESEA FINALIZAR?(S/N)";SN$
110 IF SN$="S" THEN STOP OFF:END
120 CLS:RETURN 20
130 PRINT A;"*";I;"=";R
140 RETURN
RUN
TABLAS DE MULTIPLICAR.
INGRESE UN NUMERO:? 3
TABLA DE MULTIPLICAR DEL 3
 3 * 1 = 3
 3 * 2 = 6
 3 * 3 = 9
 3 * 4 = 12
 3 * 5 = 15
 3 * 6 = 18
 3 * 7 = 21
```

```
 3 * 8 = 24
 3 * 9 = 27
 3 * 10 = 30
TABLAS DE MULTIPLICAR.
INGRESE UN NUMERO:? [CTRL] + [STOP]
DESEA FINALIZAR?(S/N)S
Ok
```

(Si hubiera contestado N habria vuelto a la linea 20. Nótese que en cualquier momento del programa, si Ud. pulsa las teclas [CTRL] + [STOP] el programa bifurca a la rutina de la línea 100).

RIGHT $

RIGHT$(expresión alfanumérica,cantidad de caracteres)

Devuelve una subcadena de caracteres de **expresión alfanumérica** cuya cantidad se especifica en **cantidad de caracteres**, contando desde el último caracter de **expresión alfanumérica**

Si **cantidad de caracteres** es mayor o igual que la cantidad de caracteres de **expresión alfanumérica**, devuelve **todos** los caracteres de dicha expresión.

Si **cantidad de caracteres=0**, devuelve una constante alfanumérica nula (**""**).

El caracter encabezamiento de gráficos (código ASCII 1) se cuenta como un caracter. Esto significa que cada caracter gráfico se cuenta por **2**.

Ejemplo:

```
10 CLS
20 INPUT "DATO=";A$
30 L=LEN(A$)
40 FOR I=1 TO L
50 B$=RIGHT$(A$,I)
60 B$=RIGHT$(SPACE$(50)+B$,L)
70 PRINT B$
80 NEXT
90 END
```

```
RUN
DATO=?TALENT
      T
     NT
    ENT
   LENT
  ALENT
 TALENT
Ok
```

RND

RND(expresión numérica)

Devuelve un número **random** (generado al azar) entre (pero no incluyendo) **Ø** y **1.**

Si el valor de **expresión numérica** es positivo:

> La misma secuencia de números aleatorios se genera cada vez que se ejecuta el programa.

Si el valor de **expresión numérica** es **Ø:**

> Repite el último número generado.

Si el valor de **expresión numérica** es negativo:

> Se generan distintas secuencias dependiendo del valor de **expresión numérica.**

Normalmente se utiliza un valor positivo para **expresión numérica.** Por lo tanto, siempre se ejecuta la misma secuencia de números aleatorios cada vez que se corre el programa, a menos que se inicialice el generador de números al azar.

Para cambiar la secuencia de números aleatorios, ejecute la sentencia **X=RND(-TIME)** para seleccionar otra secuencia, y luego utilice **RND(valor positivo).**
La variable **X** es ficticia, y puede ser cualquier número.

Ejemplo:

```
10 FOR I=1 TO 5
20 A=RND(1)
30 PRINT A;TAB(18);INT(A*10)
40 NEXT I
50 END
RUN
 .59521943994623    5
 .10658628050158    1
 .76597651772823    7
 .57756392935958    5
 .73474759503023    7
Ok
```

RUN

RUN [número de línea]

Para comenzar la ejecución de un programa.

Si se especifica **número de línea**, la ejecución comienza en esa línea. De lo contrario, la ejecución comenzará en el menor número de línea disponible en memoria.

Cuando se ejecuta el comando **RUN**, se borran todas las variables preexistentes y se cierran todos los archivos.

Ejemplo:

```
10 PRINT"DESEO"
20 PRINT "UNA COMPUTADORA"
RUN 20
UNA COMPUTADORA
Ok
```

SAVE

SAVE "dispositivo : [nombre archivo] "

Para salvar un archivo de un programa **BASIC** almacenado en la memoria en un archivo en código ASCII designado por **nombre archivo** en un dispositivo especificado por **dispositivo.**

Un programa almacenado en código ASCII es un programa en donde las líneas se almacenan en el formato de los caracteres códigos ASCII, o sea, tal cual se ven cuando se lo LISTa . Se caracteriza por lo siguiente:

Se requiere más espacio de almacenamiento que el requerido por el comando CSAVE.
Los programas que se pueden intercalar con la orden MERGE deben estar en código ASCII.
Como un programa en código ASCII puede tratarse como un archivo de datos, cada programa puede asignarse a un variable usando la instruccion LINE INPUT#.
Cada línea finaliza con la secuencia return/line feed (caracteres ASCII 13 y 10).
CTRL + Z (caracter ASCII &H1A) es tratado como fin del archivo (end-of-file).

Cuando se utiliza un archivo en cassette, la velocidad de transferencia (baudios) puede indicarse con la sentencia SCREEN.

Ejemplo:

SAVE "CAS:MUESTRA"
(Se graba el programa almacenado en memoria en un cassette bajo el nombre MUESTRA. En este caso queda grabado en formato ASCII, lo que permitirá su análisis como un archivo más (con la sentencia OPEN).
Se debe cargar este programa con la sentencia LOAD (NO CLOAD) y además se puede utilizar la sentencia MERGE para fundir este programa con otros).
Ok

SCREEN

SCREEN [modo de pantalla] [, tamaño de sprite] [, interruptor click de tecla] [, nivel baudios cassette] [, opción de impresora]

Para asignar el modo de pantalla, tamaño de los sprites, click de las teclas, nivel en baudios del cassette y la opción de una impresora.

modo de pantalla puede tomar valores de acuerdo al siguiente detalle:

Ø: 4Ø columnas x 24 filas en modo para texto (valor inicial: 37 columnas x 24 filas)
1: 32 columnas x 24 filas en modo para texto (valor inicial: 29 columnas x 24 filas)
2: modo de alta resolución
3: modo de multicolor

MODO Ø es un modo texto donde los caracteres se forman con una matriz de 5x7 puntos sobre una base de 6x8 puntos. Luego, existe un punto en blanco entre cada caracter y cada línea.
En este modo no se puede utilizar ningún comando ni sentencia para gráficos. Este modo es el modo inicial cuando se enciende la computadora.

MODO 1: es otro modo texto donde los caracteres estan formados con una matriz de 8x8 puntos. No se pueden utilizar ningún comando o sentencia para gráficos, excepto las instrucciones de manejo de sprites.

MODO 2: en el modo de alta resolución se pueden imprimir puntos de colores en una pantalla dividida en 256 puntos horizontales y 192 puntos verticales. Todas las sentencias y comandos gráficos pueden utilizarse y existen 16 colores disponibles.

MODO 3: en el modo multicolor existen 64x48 bloques de color en la pantalla de 4x4 puntos. Cada bloque puede ser de cualquiera de los 16 colores.

modo de pantalla cambia a los valores **Ø** ó **1** cuando se encuentra con una instrucción **INPUT** o el **BASIC** vuelve a su nivel comando si se estaba ejecutando un programa en modo **2** ó **3.**

tamaño de sprite determina el tamaño de los sprites definidos con la sentencia **SPRITE$,** y la magnificación de las figuras impresas en pantalla por la sentencia **PUT SPRITE.**
tamaño de sprite es una expresión entera cuyo rango es de Ø a 3 y el tamaño queda determinado conforme a lo siguiente:

Ø: celdas de configuración 8x8 puntos impresos tal cual son (8x8)
1: celdas de configuración 8x8 puntos impresos al doble de tamaño: (16x16 puntos)
2: celdas de configuración 16x16 puntos impresos tal cual son: (16x16 puntos)
3: celdas de configuración 16x16 puntos impresos al doble de tamaño: (32x32 puntos)

No bien se especifica **tamaño de sprite** el contenido de **SPRITE$** será borrado.

interruptor de click de tecla determinará si se desea habilitar o inhabilitar el click de tecla o sea, que se oiga un pequeño ruido cada vez que se pulsa una.

Ø: click de teclado desactivado
1: activa el click de teclado

En la modalidad de texto, todos los comandos gráficos generan un error: **"Illegal function call"** (llamada a función incorrecta).

nivel en baudios del cassette determina el nivel de transferencia normal en baudios (bits por segundo) para las operaciones de escritura sucesivas:

1: 12ØØ baudios
2: 24ØØ baudios.

El nivel en baudios también puede determinarse utilizando **CSAVE** con su opción de nivel en baudios.
Para la lectura de un cassette, la determinación del nivel de transferencia en baudios es automática, de manera que el usuario no tiene porque saber en cual nivel fue grabado un archivo.

opción de impresora determina si la impresora con la que se está operando es **MSX** (o sea que tiene capacidad de imprimir símbolos gráficos) o no:

Ø: la impresora **no es** MSX, los gráficos son convertidos a espacios
1: la impresora **es** MSX, por tanto dispone de capacidad gráfica.

Ejemplo:

```
SCREEN 1,2
```
(modalidad textos 32x24, sprites 16x16 no ampliados)

SGN

SGN(expresión numérica)

Permite determinar el signo de **expresión numérica** , devolviendo:

1: para **expresión numérica** mayor que Ø (positivo)
Ø para **expresión numérica** igual a Ø (cero)
-1: para **expresión numérica** menor que Ø (negativo)

Ejemplo:

```
10 CLS
20 INPUT "INGRESE UN NUMERO:";H
30 J=SGN(H)
40 J=J+2
50 ON J GOSUB 100,110,120
60 PRINT "EL SIGNO DEL VALOR DIGITADO ES:";
70 PRINT A$
80 PRINT
90 GOTO 20
100 A$="NEGATIVO":RETURN
110 A$="CERO":RETURN
120 A$="POSITIVO":RETURN
RUN
INGRESE UN NUMERO:? 7
EL SIGNO DEL VALOR DIGITADO ES:POSITIVO

INGRESE UN NUMERO:? 0
EL SIGNO DEL VALOR DIGITADO ES:CERO

INGRESE UN NUMERO:?-1
EL SIGNO DEL VALOR DIGITADO ES:NEGATIVO

INGRESE UN NUMERO:?
```

(Pulse CTRL + STOP para finalizar).

SIN

SIN(expresión numérica)

Devuelve el seno de **expresión numérica,** que debe estar expresado en radianes .

El resultado de la función **SIN** se devuelve siempre como un número real de doble precisión, sin interesar el tipo de variable numérica al que se lo asigna.

Para pasar de grados a radianes, multiplique **expresión numérica** por PI/18Ø.

Ejemplo:

```
10 FOR J=1 TO 3
20 A=SIN(J)
30 PRINT J;A
40 NEXT J
50 END
RUN
 1  .84147098480792
 2  .90929742682566
 3  .14112000805978
Ok
```

SOUND

SOUND número de registro, expresión numérica

Para enviar valores directamente a los registros del **PSG** (Programmable Sound Generator) o sea el generador programable de sonidos.

expresión numérica debe ser un valor entre **Ø** y **255.**

Existen 16 registros que utiliza el **PSG,** de los cuales 14 pueden utilizarse para cargar valores. La función de crear sonidos específicos y efectos sonoros se controlan con estos registros de la siguiente forma:

OPERACIONES	REGISTROS	FUNCION
Control generador tono	RØ a R5	Período programable de tonos.
Control generador ruido	R6	Período programable de ruido.
Control mezcla	R7	Habilita el tono o ruido en el canal correspondiente
Control amplitud	R8 a R1Ø	Selecciona amplitud "fija" o con "envolvente variable.
Control generador envolvente	R11 a R13	Período programable de envolvente y selecciona la forma de envolvente.

CONTROL GENERADOR TONO:

El **PSG** tiene 3 canales de tono: **A, B y C.** La frecuencia de cada canal se obtiene contando el ingreso de reloj interno 16 veces el valor de la frecuencia deseada.
Para obtenerlo, se debe efectuar la siguiente operacion:

Valor a Utilizar = 1789773 / (16*frecuencia)
Registro bajo = Resto de (Valor a Utilizar / 256)
Registro alto = Parte entera de (Valor a Utilizar / 256)

Estos registros bajo y alto se utilizan de a pares para cada canal:

CANAL	REGISTRO ALTO	REGISTRO BAJO
A	R1	RØ
B	R3	R2
C	R5	R4

Por ejemplo, si deseamos obtener una frecuencia de **44Ø Hertz** (un LA de concierto) en el canal A haríamos:

VU=1789773/(16*44Ø):RB=VU MOD 256: RA=INT(VU/256): SOUND ØØ,RB: SOUND 1,RA: SOUND 8,15: SOUND 7,254 [RETURN]

Se obtiene el tono en el canal **A.** Para los registros R8 y R7 lea las siguientes secciones. Para detenerlo, pulse **CTRL + STOP .**

CONTROL AMPLITUD:

Se utilizan los registros correspondientes para controlar la amplitud de los difentes canales, esto es, el volumen de cada canal. Los registros se utilizan de la siguiente forma:

Canal A: R8
Canal B: R9
Canal C: R1Ø

Cada canal tiene un volumen desde **Ø** a **15**, siendo **15** el volumen más alto.
El control de amplitud también puede utilizarse para dirigir el período de la envolvente de cada canal. Colocando un valor de **16**, la amplitud del canal correspondiente será controlada por los registros **R11, R12 y R13.** Para más información de esto, lea la sección **CONTROL GENERADOR ENVOLVENTE.**

CONTROL MEZCLA:

El registro de mezcla, o sea **R7**, controla los tres canales de Tono/Ruido. El mezclador, como se describió previamente, combina ruidos y frecuencias de tonos para cada canal.

La determinación de si se habilitan ruidos y/o tonos por cada canal se efectúa mediante el estado de los bits (digitos binarios) desde el Ø al 5 del registro **R7.**
Los bits 6 y 7 son para las puertas de **I/O** (entrada y salida) que se conectan a través del **PSG**, y son ignorados por **BASIC.**

El siguiente gráfico explica lo antedicho:

B7	B6	B5	B4	B3	B2	B1	BØ
NO USADOS		Ruido Por Canal			Tono Por Canal		
////////		C	B	A	C	B	A

Valor lógico de cada bit:
1 para canal deshabilitado
Ø para canal habilitado

Por ejemplo: **SOUND 7,&B11111111Ø** habilita el tono en canal **A** y **SOUND 7,&B1111Ø11Ø** habilita el tono y ruido en canal **A.**

CONTROL GENERADOR ENVOLVENTE:
REGISTROS CONTROL DE PERIODO DE ENVOLVENTE:

La generación de formas complejas e interesantes de envolvente puede realizarse de dos formas diferentes en **BASIC.** Primero, es posible variar la frecuencia de la envolvente mediante los registros

R11 y **R12** como un registro de **16** bits (o sea, igual que en CONTROL GENERADOR TONO); y por último, variando la forma relativa y el ciclo, mediante el registro **R13.**

Para el primer caso, la fórmula a utilizar es la misma que en el generador de tonos, siendo el **R11** el Registro Bajo y **R12** el Registro Alto.

REGISTRO CONTROL DE FORMA:

Se pueden seleccionar **9** formas de envolvente programando el registro **R13.**

Valor Utilizado	Forma
0, 1, 2, 3, 9	
4, 5, 6, 7	
8	
10	
11	
12	
13	
14	
15	

Por ejemplo: **SOUND 13,14** crea una modulación en el tono hacia arriba y hacia abajo con el período indicado por los registros **R11** y **R12,** cuando se habilita el bit 4 del registro **R8 (SOUND 8,16).**

DIAGRAMA DE BLOQUES DEL PSG:

REGISTRO \ BIT		B7	B6	B5	B4	B3	B2	B1	B0
RØ	Período Tono Canal A	Ajuste fino en tono canal A (8 Bits)							
R1						Ajuste grueso en tono canal A(4 Bits)			
R2	Período Tono Canal B	Ajuste fino en tono canal B (8 Bits)							
R3						Ajuste grueso en tono canal B(4 Bits)			
R4	Período Tono Canal C	Ajuste fino en tono canal C (8 Bits)							
R5						Ajuste grueso en tono canal C(4 Bits)			
R6	Período ruido				Ajuste grueso en ruido (5 Bits)				
R7	$\overline{\text{Enable}}$	$\overline{\text{IN/OUT}}$		$\overline{\text{Ruido}}$			$\overline{\text{Tono}}$		
		1ØB	1ØA	C	B	A	C	B	A
R8	Amplitud Canal A				M	L3	L2	L1	L0
R9	Amplitud Canal B				M	L3	L2	L1	L0
RIØ	Amplitud Canal C				M	L3	L2	L1	L0
R11	Período envolvente	Ajuste fino en envolvente (8 Bits)							
R12		Ajuste grueso en envolvente (8 Bits)							
R13	Forma envolvente/ciclo					CONT.	ATT.	ALT.	HOLD

↑
ARREGLO REGISTRADO (14 REGISTROS DE LECTURA/ESCRITURA)

Ejemplo:

```
10 CLS
20 SOUND 6,15:SOUND 7,7
30 SOUND 8,16:SOUND 9,16
40 SOUND 10,16:SOUND 11,0
50 SOUND 12,16:SOUND 13,0
60 SCREEN 3
70 OPEN "GRP:"FOR OUTPUT AS #1
80 PRESET (50,50)
90 PRINT #1,"AY!"
100 FOR I= 1 TO 1000:NEXT I:END
RUN
```

(Se oye el sonido de un disparo).

```
Ok
```

SPACE $

SPACE$ (expresión numérica)

Devuelve tantos espacios en blanco (caracter ASCII 32) como indique **expresión numérica.**

expresión numérica debe estar dentro del rango de **Ø** a **255.** Se descartan las fracciones.

Ejemplo:

```
10 OPEN "CRT:"FOR OUTPUT AS #1
20 CLS
30 FOR J= 0 TO 10
40 S=INT(RND(1)*17)
50 PRINT #1,"*";
60 PRINT #1,SPACE$(S);"*";
70 PRINT #1,SPACE$(17-S);"ESPACIO";S
80 NEXT J
90 CLOSE
100 END
RUN
*           *        ESPACIO 10
* *                  ESPACIO 1
*               *    ESPACIO 13
*          *         ESPACIO 9
*             *      ESPACIO 12
```

```
*   *                ESPACIO 3
*       *            ESPACIO 6
*                  * ESPACIO 16
*           *        ESPACIO 10
*        *           ESPACIO 7
*                *   ESPACIO 14
Ok
```

SPC

SPC(expresión numérica)

Avanza tantos espacios en la impresión como indique **expresión numérica.**

La diferencia con **SPACE$** es que ésta es una función alfanumérica, mientras que **SPC no.**

Además, **SPC** sólo puede utilizarse con las declaraciones **PRINT** y **LPRINT.**
expresión numérica debe encontrarse dentro del rango de **Ø a 255.**

Ejemplo:

```
10 CLS
20 FOR J= 0 TO 10
30 S=INT(RND(1)*17)
40 PRINT "*";
50 PRINT SPC(S);"*";
60 PRINT SPC(17-S);"ESPACIO";S
70 NEXT J
80 END
RUN
*           *        ESPACIO 10
* *                  ESPACIO 1
*               *    ESPACIO 13
*          *         ESPACIO 9
*             *      ESPACIO 12
*   *                ESPACIO 3
*       *            ESPACIO 6
*                  * ESPACIO 16
*           *        ESPACIO 10
*        *           ESPACIO 7
*                *   ESPACIO 14
Ok
```

SPRITE ON/OFF/STOP

Para activar o desactivar la detección de colisión entre sprites durante la ejecución de un programa **BASIC.**

Una instrucción **SPRITE ON** activa la detección de colisión entre sprites.

Si se especifica un número de línea en el comando **ON SPRITE GOSUB**, cada vez que el **BASIC** comienze un nuevo comando, verificará la colisión de sprites. Si esto sucede, ejecutará el **GOSUB** a la línea especificada.

Si ha sido ejecutado un **SPRITE OFF** no tiene lugar ninguna detección y no se memorizará el evento aunque tenga lugar.

En cambio si se ha ejecutado un **SPRITE STOP**, no tendrá lugar ninguna detección, pero si los sprites coinciden, esto es memorizado y cuando un **SPRITE ON** sea ejecutado, tendrá lugar una inmediata detección.

Ejemplo:

Ver el ejemplo de ON SPRITE GOSUB.

SPRITE $

SPRITE$ (expresión entera)=expresión alfanumérica

Es la forma de definir el diseño de los sprite.

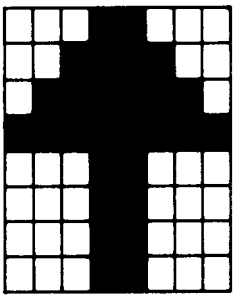
expresión entera debe ser menor de **256** cuando el tamaño de los sprites es **Ø** o **1** y menos de **64** cuando el tamaño sea **2** o **3.**

El contenido de **SPRITE$** esta fijado en **32** bytes, de manera que si el largo de **expresión alfanumérica** es menor a **32** caracteres, son los mismos son completados con **CHR$(Ø).**

Cada caracter de **expresión alfanumérica** responde al patrón de una fila del diseño, quedando este determinado, si es en binario, **1** para punto encendido y **Ø** para punto apagado.

expresión alfanumérica para tamaño 8x8 puntos:

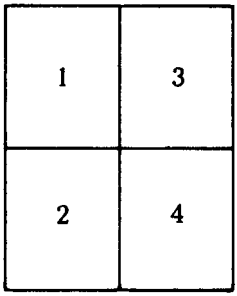
Cada fila (comprendiendo 8 puntos) de la matriz de la figura sprite se representa con un patrón de bits.
Se utilizan 8 caracteres para representar la matriz completa del sprite.



```
CHR$(&B00011000)
CHR$(&B00111100)
CHR$(&B01111110)
CHR$(&B11111111)
CHR$(&B00011000)
CHR$(&B00011000)
CHR$(&B00011000)
CHR$(&B00011000)
```

expresión alfanumérica para tamaño 16x16 puntos:

La figura del sprite se divide en 4 secciones de 8 x 8 puntos cada una, y estas secciones quedan ubicadas como en la figura. Dado que la matriz de cada sección se representa con 8 caracteres, se requieren 32 caracteres para representar las cuatro matrices.



Es recomendable recodificar estos patrones binarios en otra base (p.ej. decimal o hexadecimal) para ahorrar memoria, tiempo y esfuerzo.

Ejemplo:

```
10 SCREEN 1,2
20 PRINT "***RATON***"
30 FOR J=1  TO 16
40 READ D$
50 A$=A$+CHR$(VAL("&B"+LEFT$(D$,8)))
60 B$=B$+CHR$(VAL("&B"+RIGHT$(D$,8)))
70 NEXT J
80 SPRITE$(0)=A$+B$
90 PUT SPRITE 0,(50,70),15,0
100 PUT SPRITE 1,(90,70),14,0
110 PUT SPRITE 2,(130,70),1,0
120 PUT SPRITE 3,(170,110),13,0
130 PRINT "PUT SPRITE 0,(0,208)
140 DATA 0000000000011110
150 DATA 0000100000101001
160 DATA 0001011111101101
170 DATA 0000100000101001
180 DATA 0011111011111111
190 DATA 0001111111111000
200 DATA 0000001111111000
210 DATA 0000011111110000
220 DATA 0000001111100010
230 DATA 0000000111100100
240 DATA 1100001111100100
250 DATA 0011111111110010
260 DATA 0000001111110010
270 DATA 0000001111110010
280 DATA 0000000111111100
290 DATA 0000011111110000
RUN
```

(Se ven tres ratones en la pantalla. Para apagarlos, aproveche lo impreso en la pantalla:**PUT SPRITE Ø,(Ø,2Ø8)**)

SQR

SQR(expresión numérica)

Devuelve la raíz cuadrada de **expresión numérica** y se calcula internamente en doble precisión.

expresión numérica debe ser mayor que **Ø**

Ejemplo:

```
10 INPUT "J=";J
20 PRINT "RAIZ DE J=";
30 PRINT SQR(J)
40 PRINT" = J^0.5 =";
50 PRINT J^.5
60 PRINT
70 GOTO 10
RUN
J=? 23
RAIZ DE J= 4.7958315233127
 = J^0.5 = 4.7958315233124

J=? 55
RAIZ DE J= 7.4161984870955
 = J^0.5 = 7.4161984870947

J=?
```

(Pulse [CTRL] + [STOP] para finalizar).

STICK

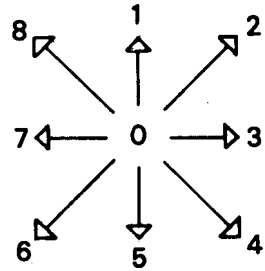
STICK(expresión entera)

Devuelve la dirección de operación del joystick especificado por **expresión entera,** o cual tecla de cursor fue presionada.

expresión entera puede variar de **Ø** a **2,** especificando el joystick o los cursores que van a ser verificados.

Ø : los cursores son utilizados como joystick.
1 : podrá utilizarse el joystick conectado a la puerta 1.
2 : podrá utilizarse el joystick conectado a la puerta 2.

El resultado devuelto es un entero de **1** a **8,** representando la dirección de operación. Cuando no se opera el joystick, la función devuelve cero.



Cuando dos cursores de direcciones ortogonales se pulsan simultáneamente, el valor que devuelve **STICK** es el indicado para las direcciones diagonales.
Por ejemplo, si pulsamos simultáneamente el cursor ↑ y el cursor → , la función **STICK** devuelve **2.**

Ejemplo:

```
10 CLS
20 LOCATE 0,10
30 PRINT "CURSORES";TAB(10);"JOYSTICK 1";
TAB(22);"JOYSTICK 2"
40 PRINT TAB(3);STICK(0);
50 PRINT TAB(12);STICK(1);
60 PRINT TAB(24);STICK(2)
70 GOTO 20
RUN
CURSORES   JOYSTICK 1   JOYSTICK 2
 . 0          0            0
```

(Los números respectivos varían según se pulse la tecla correspondiente. Para que Joystick 1 o 2 den distinto de cero, debe estar colocado el Joystick correspondiente. Detener con [CTRL] + [STOP].

STOP

Para suspender la ejecución de un programa y volver al nivel de comando.

El comando **STOP** puede ser utilizado en cualquier parte de un programa para suspender su ejecución.

Cuando se encuentra una instrucción STOP, se imprime el siguiente mensaje:

Break in nnn (siendo **nnn** un número de línea)

A diferencia con la instrucción **END**, el **STOP** no cierra los archivos y se puede reanudar la ejecución mediante el comando **CONT.**

Ejemplo:

```
10 INPUT "INGRESE NRO. (1 O 2)";N
20 IF N=1 THEN PRINT "END":END
30 IF N=2 THEN PRINT "STOP":STOP
40 GOTO 10
RUN
INGRESE NRO. (1 O 2) 2
STOP
Break in 30
Ok
Cont
INGRESE NRO. (1 O 2) 1
END
Ok
```

STOP ON/OFF/STOP

Para activar o desactivar la detección de un **CTRL + STOP** durante la ejecución de un programa **BASIC.**

Un **STOP ON** debe ser ejecutado y activa así la detección de un **CTRL + STOP.**

Si se especifica un número de línea en el comando **ON STOP GOSUB,** cada vez que el **BASIC** comienze un nuevo comando, verificará si se pulsó **CTRL + STOP.** Si esto sucede, ejecutará el **GOSUB** a la línea especificada.

Si se hubiera ejecutado un **STOP OFF,** no tendrá lugar ninguna detección y el evento no es memorizado aunque tenga lugar.

En cambio si ha sido ejecutado un **STOP STOP,** no tendrá lugar ninguna detección, pero si se tipeó un **CTRL + STOP** éste será memorizado de manera que cuando se ejecute un **STOP ON** tendrá lugar una inmediata detección.

Ejemplo:

Ver el ejemplo de ON STOP GOSUB.

STR $

STR$(expresión numérica)

Devuelve una cadena alfanumérica que representa a **expresión numérica.**

Esto permite tratar a la cadena alfanumérica como tal, o sea que se le pueden aplicar todas las funciones y operaciones alfanuméricas.

expresión numérica puede ser de cualquier tipo.

Ejemplo:

```
10 A=1957:B=3:C=6:CLS
20 A$=STR$(A)+STR$(B)+STR$(C)
30 PRINT A$;" TOTAL DE CARACTERES:";LEN(A$)
40 PRINT A+B+C
50 END
RUN
 1957 3 6 TOTAL DE CARACTERES: 9
 1966
Ok
```

STRIG

STRIG (expresión entera)

Devuelve el estado del botón disparador de un joystick.

expresión entera puede estar dentro del rango de **Ø** a **4**, actuando de acuerdo al siguiente detalle:

- **Ø:** se usa la barra espaciadora como disparador.
- **1:** se usa el primer botón disparador del joystick conectado en la puerta 1.
- **2:** se usa el primer botón disparador del joystick conectado en la puerta 2.
- **3:** se usa el segundo botón disparador del joystick conectado en la puerta 1.
- **4:** se usa el segundo botón disparador del joystick conectado en la puerta 2

Si el disparador no está siendo apretado devuelve Ø y de lo contrario devuelve -1.

Ejemplo:

```
10 PRINT STRIG(0);
20 PRINT STRIG(1);
30 PRINT STRIG(2)
40 GOTO 10
RUN
 0  0  0
 0  0  0
-1 -1  0
-1  0  0
 0 -1  0
-1  0  0
-1 -1  0
 0  0  0
 0  0  0
```

(En este caso, se pulsó la barra espaciadora y el disparador del Joystick 1. Pulse [CTRL] + [STOP] para finalizar).

STRIG ON/OFF/STOP

STRIG(expresión entera) ON
STRIG(expresión entera) OFF
STRIG(expresión entera) STOP

Para activar y desactivar la detección del pulsado de los botones disparadores de los joysticks en un programa **BASIC.**

expresión entera puede estar dentro del rango de **Ø a 4.**

Para detalles de **expresión entera** ver la instrucción **STRIG.**

Se deberá ejecutar un **STRIG(expresión entera) ON** para activar la captura de los disparadores.

Especificando un número de línea después del comando **ON STRIG GOSUB,** cada vez que el **BASIC** comience una nueva instrucción, verificará si el botón disparador ha sido seleccionado y de ser asi ejecutará un **GOSUB** a la línea indicada.

Si ha sido ejecutado un **STRIG (expresión entera) OFF**, no tendrá lugar ninguna detección y el evento no será memorizado aunque tenga lugar.

En cambio, si ha sido ejecutado un **STRIG (expresión entera) STOP**, no se activa ninguna captura pero si se pulsa el disparador, esta acción será memorizada, y una detección inmediata tendrá lugar cuando se ejecute **STRIG (expresión entera) ON.**

Ejemplo:

Ver ejemplo en ON STRIG GOSUB.

STRING$

STRING$(expresión entera, expresión alfanumérica)
STRING$(expresión entera, código de caracter)

Devuelve una cadena alfanumérica de largo indicado por **expresión entera,** formado por caracteres cuyo código **ASCII** es tomado del primer caracter de **expresión alfanumérica** o de **código de caracter.**

El valor de **expresión entera** está en el rango de **Ø** a **255.**

Sólo el primer caracter de **expresión alfanumérica** se tiene en cuenta; el segundo y los restantes son ignorados.

El caracter encabezamiento de gráficos (código ASCII **1**) se cuenta como un caracter. Luego, si se coloca en el primer lugar de **expresión alfanumérica** un símbolo gráfico, **STRING$** devuelve un valor alfanumérico formado por caracteres código **1.**

Ejemplo:

```
10 CLS
20 FOR J=3 TO 13
30 L=INT(RND(1)*20)
40 LOCATE 3,J
50 PRINT USING "##";L
60 LOCATE 6,J
70 PRINT STRING$(L,"*")
80 NEXT J
90 END
RUN
```

```
 11 ***********
  2 **
 15 ***************
 11 ***********
 14 **************
  3 ***
  7 *******
 18 ******************
 12 ************
  9 *********
 16 ****************
Ok
```

SWAP

SWAP variable 1, variable 2

Para intercambiar el valor de dos variables.

Cualquier tipo de variable puede ser intercambiada (entero, simple precisión, doble precisión alfanumérica) pero las dos variables deberán ser del mismo tipo, caso contrario producirá un error con un mensaje: **"Type mismatch"**(no concuerda el tipo de variable).

Ejemplo:

```
10 FOR J=0 TO 3
20 X(J)=INT(RND(1)*99)
30 Y(J)=INT(RND(1)*99)
40 NEXT J
50 GOSUB 120
60 PRINT
70 FOR J=0 TO 3
80 SWAP X(J),Y(J)
90 NEXT J
100 GOSUB 120
110 END
120 PRINT "J    X(J)    Y(J)"
130 PRINT
140 FOR J=0 TO 3
150 PRINT USING"#    ##      ##";J;X(J);Y(J)
160 NEXT J
170 RETURN
RUN
```

```
J    X(J)    Y(J)

0    58      10
1    75      57
2    72      18
3    36      94

J    X(J)    Y(J)

0    10      58
1    57      75
2    18      72
3    94      36
Ok
```

TAB

TAB(expresión entera)

Determina la columna de comienzo de la próxima impresión cuando se utilizan los comandos **PRINT y LPRINT.**

Si la última posición impresa es superior al valor de **expresión entera, TAB** no afecta la impresión.

TAB(Ø) es la columna extrema izquierda, mientras que la extrema derecha vale el ancho dado por **WIDTH-1.**

expresión entera debe estar dentro del rango de **Ø a 255.**

Ejemplo:

```
10 PRINT
20 PRINT"*",TAB(9)"*"
30 PRINT"*","*"
40 PRINT "*"TAB(9)"*"
50 PRINT "*";TAB(9);"*"
60 END
RUN
*               *
*               *
*         *
*         *
Ok
```

TAN

TAN(expresión numérica)

Devuelve la tangente de **expresión numérica,** que debe estar expresada en radianes.

TAN se calcula con doble precisión.

Si **TAN** se desborda, el error **"Overflow"** correspondiente será emitido.

Ejemplo:

```
10 FOR I=3.14 TO 6.28
20 A=TAN(I*3.14)
30 PRINT A
40 NEXT
RUN
 .46447027876367
 .46253546743286
 .46060350459682
 .45867437419176
Ok
```

TIME

Es un sistema de medición de tiempo interno.

TIME se incrementa automáticamente en **1** cada vez que el **VDP** genera una interrupción (**5Ø** veces por segundo).

Es posible inicializar la variable **TIME** mediante la instrucción **TIME=Ø**

El contador de **TIME** se detiene cuando se inhabilitan las interrupciones de **VDP** (por ejemplo, cuando se utiliza el grabador)

Ejemplo:

```
10 REM CRONOMETRO
20 CLS
30 LOCATE 10,8
40 PRINT "comienzo!"
50 TIME=0
60 LOCATE 10,10
70 T=TIME/50
80 H=INT(T/3600)
90 M=INT(T/60)
100 S=T MOD 60
110 PRINT USING"## hs ## min ## seg";H;M;S
120 GOTO 60
RUN
comienzo!
            0 hs  0 min 23 seg
```

(Es un cronómetro preciso. Para detenerlo, pulse [CTRL] + [STOP].

TRON/TROFF

TRON
TROFF

Para hacer el seguimiento de la ejecución de las líneas de un programa.

Como una ayuda para la depuración (debugging), el comando **TRON** (ejecutado bien sea en el modo directo o en el indirecto) habilita una bandera de seguimiento que imprime cada línea del programa a medida que es ejecutada.
Los números aparecen incluídos entre corchetes.
La bandera de seguimiento es inhabilitada con el comando **TROFF** (o cuando se ejecuta la instrucción **NEW** (borrar programa))

Ejemplo:

```
10 PRINT"computadora ";
20 PRINT"msx ";
30 PRINT "TALENT"
RUN
computadora msx TALENT
OK
```

```
TRON
Ok
RUN
[10]computadora [20]msx [30]TALENT
OK
TROFF
Ok
RUN
computadora msx TALENT
Ok
```

USR

USR [número] (argumento)

Ejecuta desde la posición de memoria especificada en **DEFUSR** una subrutina del lenguaje assembler

argumento es el nexo de variables entre el BASIC y el Lenguaje de Máquina. Si no se utiliza, poner **USR(Ø).**

número está dentro del rango de **Ø a 9** y corresponde al dígito provisto con la declaración **DEFUSR** para dicha rutina. Si se omite **número, USR Ø** es asumido.

Ejemplo:

```
10 CLS:SCREEN 1
20 FOR I=0 TO &H2A0
30 A=RND(1)*100+33
40 PRINT CHR$(A);
50 POKE &H9000+I,A
60 NEXT I:POKE &H9000+I,0
70 FOR I=0 TO 11
80 READ D$:D$="&H"+D$:D=VAL(D$)
90 POKE &H9400+I,D
100 NEXT I:BEEP
110 IF INKEY$="" THEN 110
120 CLS:DEF USR=&H9400
```

```
130 A=USR(0)
140 GOTO 140
150 DATA 11,00,90,1A,A7,C8,CD,A2,00,13,18,F7
RUN
```

(Se puede apreciar la diferencia de velocidades en la ejecución entre BASIC y lenguaje de máquina. Pulse cualquier tecla luego de escuchar un Bip. Pulse [CTRL] + [STOP] para finalizar).

VAL

VAL(expresión alfanumérica)

Devuelve el valor numérico de la **expresión alfanumérica**, si está formada por caracteres de números. La función VAL también toma espacios en blanco (blanks) para los comienzos, tabuladores y alimentación de línea(line feed) desde **expresión alfanumérica**

expresión alfanumérica puede ser binario **(&B)**, octal **(&O)**, o hexadecimal **(&H)**, ya que también representan números.

Si **expresión alfanumérica** no es numérica (posee letras por ejemplo), VAL devuelve **Ø**

Ejemplo:

```
10 FOR I=0 TO 30 STEP 5
20 PRINT "HEX ";HEX$(I);TAB(7);"ES";VAL("&H"+
HEX$(I))
30 NEXT I
40 END
RUN
HEX 0  ES 0
HEX 5  ES 5
HEX A  ES 10
HEX F  ES 15
HEX 14 ES 20
HEX 19 ES 25
HEX 1E ES 30
Ok
```

VARPTR

VARPTR(nombre de variable)
VARPTR(# número de archivo)

Devuelve la dirección de memoria donde está ubicado el primer byte de datos identificados como **nombre de variable.**

Debe asignarse un valor al **nombre de variable** antes de ejecutar un **VARPTR,** de lo contrario resulta un error con su mensaje: **"Illegal function call"** (Llamado a función incorrecto).

Se puede utilizar cualquier tipo de variable (numérica, alfanumérica, matriz) y la dirección devuelta será un número entero dentro del rango de -32768 hasta 32767. Si se devuelve una dirección negativa, súmela a 65536 para obtener la dirección verdadera.

VARPTR es usualmente utilizada para obtener la dirección de una variable o de una matriz , (array) de manera de poder pasarla a una subrutina en lenguaje de máquina.

Una llamada a la función con la forma: **VARPTR(A(Ø))** es especificada usualmente cuando se pasa una matriz, de manera que la dirección más baja de la misma es la devuelta.

Todas las variables simples deberían estar definidas antes de utilizar **VARPTR** con una matriz, porque la dirección de memoria de las matrices cambia siempre que se define una nueva variable simple.

Si se especifica **número de archivo, VARPTR** devuelve la dirección del comienzo del bloque de control del archivo.

Ejemplo:

```
PRINT HEX$(VARPTR(A))
```

(Devuelve la dirección de memoria donde comienzan los datos de la variable **A.**)

VDP

VDP(expresión entera)

Si **expresión entera** se halla dentro del rango de **Ø a 7,** devuelve el valor almacenado en dicho registro de escritura solamente de la **VDP.** Si **expresión entera** es **8,** devuelve el valor almacenado en el registro de estado de la **VDP** y es sólo para lectura (read only).

Los registros de la **VDP TMS 9918A/9929A** son los siguientes:

REG#	BIT#	DESCRIPCION
Ø	Ø-5	Reservados (deben ser ceros)
	6	M3 (bit Modo 3): **1=Modo Gráficos II** (alta resolución)
	7	Habilitación VDP externa: **Ø=no/ 1=si**
1	Ø	Selección 4/16k VDP RAM: **Ø=4k/ 1=16k**
	1	Habilitación pantalla: **Ø=apaga/ 1=enciende**
	2	Habilitación interrupción: **Ø=no/ 1=si**
	3	M1 (bit Modo 1): **1= Modo texto (4Ø x 24)**
	4	M2 (bit Modo 2): **1= Modo multicolor**
	5	Reservado
	6	Tamaño: selección tamaño sprite: **Ø=8x8 bit 1=16x16bit**
	7	Magnificación:**Ø=1x/1=2x**

M1	M2	M3	MODO
Ø	Ø	Ø	32 columnas
Ø	Ø	1	alta resolución
Ø	1	Ø	multi-color
1	Ø	Ø	4Ø columnas

2	Ø-3	Reservados
	4-7	Posición de memoria de la tabla de pantalla Se obtiene así: **R2*&H4ØØ**
3	Ø-7	Posición de memoria de la tabla de color. Se obtiene así: **R3*&H4Ø**
4	Ø-4	Reservados
	5-7	Posición de memoria de tabla de diseños. Se obtiene así: **R4*&H8ØØ**
5	Ø	Reservado
	1-7	Posición de memoria de tabla de atributos de sprite. Se obtiene así: **R5*&H8Ø**
6	Ø-4	Reservados
	5-7	Posición de memoria de tabla de diseños de sprite. Se obtiene así: **R6*&H8ØØ**

7	Ø-3	Color frente (color del caracter) en modo texto
	4-7	Color fondo (color de pantalla) en todos los modos.
Estado	Ø	Bit indicador de interrupción.
	1	5 o más sprites en la misma fila.
	2	Coincidencia de sprites.
	3-7	Número del 5to. sprite cuando el bit 1 está encendido (=1)

Nota: los bits se cuentan de: Ø-7 de izquierda a derecha (el más significativo=bit Ø)

Ejemplo:

```
SCREEN 1,0:PRINT BIN$(VDP(1)) RETURN
11100000
Ok
```

(Vemos que están seleccionados: 16 k VDP RAM, pantalla encendida, interrupción habilitada, M1 y M2 deshabilitados, bit 5=Ø (reservado),tamaño sprite 8x8bit, magnificación=1x)

VPEEK

VPEEK(dirección memoria VDP)

Regresa el valor almacenado en la memoria del procesador de pantalla **(VDP). dirección memoria VDP** deberá estar en el rango de **Ø a 16383.**

Ejemplo:

```
10 CLS:SCREEN 1:KEYOFF
20 INPUT"INGRESE,CARACTER:";A$:PRINT
30 IF A$="FIN" THEN END
40 AI=ASC(A$)*8:AF=AI+7
50 PRINT "POSICION MEMORIA INICIAL: ";AI
60 PRINT "POSICION MEMORIA FINAL:   ";AF
70 PRINT "PATRON EN MEMORIA VDP":PRINT:PRINT
80 FOR I=AI TO AF
90 PRINT RIGHT$("00000000"+BIN$(VPEEK(I)),8)
100 NEXT:PRINT:PRINT:PRINT:GOTO 20
RUN
```

(Permite analizar los patrones de caracteres almacenados en la memoria de VDP. Digite FIN para finalizar).

VPOKE

VPOKE dirección memoria VDP, expresión entera

Para enviar un byte (valor entre **Ø y 255**) dado por **expresión entera** a la **dirección memoria VDP** especificada.

dirección memoria VDP debe hallarse en el rango de **Ø a 16383.**

El uso erróneo de esta instrucción puede causar pantallas "anormales". Antes de usar esta instrucción, asesórese sobre la **VDP** con los manuales **MSX** al respecto.

Ejemplo:

```
10 DEFINT I-K
20 SCREEN 1
30 FOR I= 1 TO 50
40 LOCATE RND(1)*32,RND(1)*22:PRINT CHR$(RND
(1)*200+32)
50 NEXT I
60 I=8192+RND(1)*16:K=RND(1)*255
70 VPOKE I,K
80 IF INKEY$<>"A" THEN 60
90 SCREEN1:END
```

RUN

(Aparecen letras al azar y comienzan a cambiar de color en forma individual. Hay que notar que esta opción no está disponible en **BASIC MSX.** Pulse **A** para finalizar).

WAIT

WAIT dirección puerta entrada, expresión entera 1 [, expresión entera 2]

Suspende la ejecución del programa mientras verifica el estado de la puerta de entrada indicada.

Con el dato leído en dicha puerta se efectúa la operación **XOR** (or exclusivo) con el valor especificado en **expresión entera 2** y luego la operación **AND** (y) con la expresión entera en **expresión entera 1.**

Si el resultado es cero, **BASIC** sigue verificando el dato leído, cas contrario el programa continúa con la siguiente línea. Si se omite **expresión entera 2**, se supone valor cero.

Ejemplo:

```
WAIT 1,&H22,&H22
```

WIDTH

WIDTH expresión entera

Permite seleccionar el total de caracteres que se pueden imprimir en un reglón.

Para modo de pantalla **Ø** (o sea **SCREEN Ø expresión entera** debe estar en el rango entre **1** y **4Ø**, mientras que para modo de pantalla **1** (o sea **SCREEN 1**) debe estar entre **1** y **32.**

Los valores iniciales cuando se utiliza la sentencia **SCREEN** son:

SCREEN Ø: 37 columnas.
SCREEN 1: 29 columnas.

Ejemplo:

```
10 FOR I= 5 TO 30 STEP 5
20 WIDTH I
30 FOR J=1 TO 24
40 PRINT STRING$(I,"*")"WIDTH";I
50 NEXT J:FOR J=0 TO 500:NEXT J
60 NEXT I
70 GOTO 10
RUN
```

(Van cambiando los anchos de pantalla. Pulse [CTRL] + [STOP] para finalizar).

Programas de ejemplo

Editor de Sprites

Introducción

Para cualquier programador de juegos, los gráficos son esenciales. Si Ud. es un aficionado a este tipo de programas, ya se habrá dado cuenta el problema que surge cuando se quieren diseñar sprites. Basta con equivocarse en un dato y una simpática ranita se transforma en un monstruo deforme...

Este programa le permite evitar la complicada codificación de sprites, generando incluso los datos en un archivo (que puede estar en cassette, diskette, quick disk, etc.) que luego se puede cargar como un programa más o incorporar a otro con la instrucción **MERGE**.

Características

1. Color, gráficos y sonido se aprovechan para facilitar el manejo.
2. El programa se maneja con opciones (menú).
3. Se aprovecha el manejo por interrupciones del teclado para menú
4. Admite magnificación 0, 1 y 2,3 para generar datos.
5. Los datos grabados pueden cargarse o fundirse con otro programa

Utilización

1. Carge el programa, copiándolo tal como se encuentra. Para mayor claridad hemos utilizado un programa listador que deja sangría (espacios) al principio de cada renglón de un mismo número de línea. **NO** es necesario que Ud. copie estos espacios.

2. Utilice las flechas para desplazar al cursor.

3. Con la barra espaciadora, cambia de tipo de cursor (rellena o borra).

4. Cuando está conforme con el diseño, vaya al paso 7.

5. Las opciones con las teclas de función son las siguientes:
   F1: BLANCO. Define el color de su tablero en blanco (apagado)
   F2: NEGRO. Define el color de su tablero en negro (encendido)
   F3: ITEMS. Estas son más opciones. Vaya al paso 6
   F4: FIN. Finaliza el programa
   F5: DATOS. Carga el dibujo codificado en la línea 50 en su

tablero. Si quiere cargar un diseño generado por Ud. en su tablero, siga los siguientes pasos:

(1) Carge su programa Generador de Sprites.
(2) Digite MERGE "nombre" (si está en cassette) o como indique el dispositivo.
(3) Encuentre la línea donde está el diseño. Trate de que esté ubicado en la línea 50 o menores.
(4) Mueva esta línea a la 50.
(5) Ejecute el programa
(6) Pulse F5 para DATOS (su diseño debe aparecer en pantalla)

NOTA: Sus datos deben estar almacenados en hexadecimal sin "&H" y deben ser 32. Si son menos, completar con ceros.

6. Las opciones adicionales son las siguientes:

   A. ESPEJO V cambia el dibujo en el tablero como si estuviera reflejado en un espejo ubicado en forma vertical en un costado de la figura.
   B. ESPEJO H cambia el dibujo en el tablero como si estuviera reflejado en un espejo ubicado en forma horizontal en la parte superior (o inferior) de la figura.
   C. INVERSO invierte el color de la figura.
   D. ROTAR gira la figura en sentido horario.
   E. MOVER mueve la figura un espacio hacia cualquiera de estas direcciones: arriba, abajo, izquierda y derecha.
   F. INICIO vuelve a la pantalla principal.

7. Cuando finalice el dibujo, pulse **RETURN.**

8. Se define el sprite. Si desea grabar los datos, pulse G. Si no cualquier tecla.

9. Si presiona G, aparece la siguiente opción:

   1. MAG 0,1 (tamaño sprite 0,1)
   2. MAG 2,3 (tamaño sprite 2,3)

   Seleccione su opción (ver **SPRITE$** y **PUT SPRITE** para detalles en el tamaño de sprite)

   Luego de ésto, se le pregunta por el nombre de archivo y número de línea.
   Los nombre de archivo que utilice dependen del dispositivo. Por ejemplo, si utiliza cassette debe colocar "CAS:", si es quick disk , "QD:". Luego coloque el nombre del archivo. Con diskettes no es necesario especificar el dispositivo.

Si se equivoca con el nombre de archivo, indicar línea No. 66000.

10. Luego de grabar, se sigue con el paso 2.

Aclaración: en caso de error, puede llegar a perder los datos. Trate de evitarlos fundamentalmente en el archivo. Recuerde que si utiliza cassette, debe pulsar el botón de grabación de su grabador antes de ingresar el número de línea.

Para utilizar los datos generados, deberá leerlos con el siguiente formato de programa:

```
10 SCREEN 1,2:FORI=1TO32
20 READA$:A=VAL("&H"+A$)
30 SP$=SP$+CHR$(A):NEXT
40 SPRITE$(0)=SP$
50 PUT SPRITE0,(100,100)
60 GOTO60
70 'Aqui van los datos
```

Si selecciona tamaño 0 o 1, el FOR-NEXT es hasta 8.

Listado

```
10 '***********************
20 '* GENERADOR DE SPRITES*
30 '***********************
40 '++CARGUE DATOS ANTES++
50 DATA 3F,40,40,40,5B,5B,FF,80,BF,A0,AA,A0,BF,80,
   80,FF,FC,2,2,2,62,6A,FF,55,FF,D5,FF,D5,FF,55,7F
   ,FF
60 '++O EN LA LINEA 50 CON 'MERGE'++
70 '*** INICIALIZACION ***
80 SCREEN 1,3:WIDTH 30:KEY OFF:CLS: DEFINT I-L:RES
   TORE 100
90 FOR I=1 TO 5:READ A$:KEY I,A$:NEXT:KEY ON:CLEAR
    512
100 DATA BLAN,NEGR,ITEM,FIN,DATO
110 DIM GP$(16),B(8,8),SP(16,16):T=21:K=100:C=100:
    LX=4:LY=4
120 CH=100:VP=0:FK=0:GOSUB 1080:CH=101:VP=255:GOSU
    B 1080
130 FK=1:CH=102:RESTORE 140:GOSUB 1080:CH=103:GOSU
    B 1080
```

```
140 DATA 0,7E,42,42,42,42,7E,0,FF,81,BD,BD,BD,BD,8
    1,FF
150 VPOKE 8195+9,&H1F
160 '*** PANTALLA ***
170 CLS:SPRITE$(0)=STRING$(64,255):PUT SPRITE0,(19
    1,10)
180 CLS:LOCATE 4,0:PRINT "GENERADOR DE SPRITES  ":
    LOCATE 2,3:PRINT"  1234567812345678"
190 FOR I=0 TO 1:FOR I1=1 TO 8:LOCATE 3,3+I1+8*I:P
    RINT CHR$(48+I1):NEXT I1:NEXT
200 FOR I=4 TO 19:LOCATE 4,I:PRINT STRING$(16,CHR$
    (K)):NEXT I:C=K:LOCATE LY,LX:PRINT CHR$(C+2)
210 '*** INGRESO POR TECLAS ***
220 ON STRIG GOSUB 1120:STRIG(0)ON
230 ON KEY GOSUB 360,370,410,380,1020:FOR I=1 TO 5
    :KEY(I)ON:NEXT
240 FOR I=1 TO 5:KEY(I)ON:NEXT:STRIG(0)ON
250 IF INKEY$=CHR$(13) THEN GOSUB 790
260 K=STICK(0):X=0:Y=0:ON K+1 GOTO 240,270,280,290
    ,300,310,320,330,340
270 X=-1:GOTO 750
280 X=-1:Y=1:GOTO 750
290 Y=1:GOTO 750
300 X=1:Y=1:GOTO 750
310 X=1:GOTO 750
320 X=1:Y=-1:GOTO 750
330 Y=-1:GOTO 750
340 X=-1:Y=-1:GOTO 750
350 LOCATE LY,LX:PRINT CHR$(C+2):GOTO 240
360 K=100:RETURN 200
370 K=101:RETURN 200
380 CLS
390 IF INKEY$<>"" THEN 390 ELSE SCREEN 0:DEFUSR=&H
    3E:A=USR(0):CLS:END
400 '*** OPCIONES ***
410 FOR I=1 TO 5:KEY(I)OFF:NEXT:STRIG(0)OFF
420 LOCATE T,6:PRINT "1 ESPEJOV":LOCATE T,8:PRINT
    "2 ESPEJOH":LOCATE T,10:PRINT "3 INVERSO"
430 LOCATE T,12:PRINT "4 ROTAR ":LOCATE T,14:PRINT
     "5 MOVER":LOCATE T,16:PRINT "6 INICIO"
440 LOCATE T,18
450 A$=INPUT$(1):IF A$<"1" OR A$>"6" THEN 450 ELSE
     PLAY"O5C16":LOCATE LY,LX:H=VAL(A$):PRINT CHR$
    (C):LOCATE 22,H*2+4:PRINT CHR$(42)
460 IF H=6 THEN 690 ELSE IF H=3 THEN 720 ELSE IF H
    <>5 THEN 610
470 '** MOVER **
```

```
480 LOCATE T,6:PRINT "1 ARRIBA ":LOCATE T,8:PRINT
    "2 ABAJO  "
490 LOCATE T,10:PRINT "3 IZQUIER":LOCATE T,12:PRIN
    T "4 DERECHA":LOCATE T,14:PRINT STRING$(9,32):
    LOCATE T,16:PRINT STRING$(9,32)
500 LOCATE T,14
510 A$=INPUT$(1):IF A$<"1" OR A$>"4" THEN 510 ELSE
     LOCATE 22,VAL(A$)*2+4:PRINT CHR$(42)
520 '* ARRIBA,ABAJO *
530 BEEP: IF A$>"2" THEN 570 ELSE IF A$="2" THEN K
    =20:S=-1 ELSE K=5:S=1
540 FOR I=K TO S*15+K STEP S:FOR J=5 TO 20:H=VPEEK
    (6145+(I+S-1)*32+J-1):IF H<>100 AND H<>101 THE
    N H=100
550 LOCATE J-1,I-1:PRINT CHR$(H):NEXT J,I:GOTO 690

560 '* DERECHA,IZQUIERDA *
570 IF A$="3" THEN K=5:S=1 ELSE K=20:S=-1
580 FOR J=K TO S*15+K STEP S:FOR I=5 TO 20:H=VPEEK
    (6145+(J+S-1)+(I-1)*32):IF H<>100 AND H<>101 T
    HEN H=100
590 LOCATE J-1,I-1:PRINT CHR$(H):NEXT I,J:GOTO 690

600 '** OBTIENE DIBUJO **
610 FOR I=0 TO 15:FOR J=0 TO 15:SP(I+1,J+1)=VPEEK(
    6145+(I+4)*32+J+4):NEXT J:VPOKE 6145+(I+4)*32+
    20,42:NEXT I
620 FOR I=3 TO 19:VPOKE 6145+(I+1)*32+20,32:NEXT
630 '** ESPEJOS **
640 IF H=1 THEN K=4 :S=-21 ELSE IF H=2 THEN K=-21:
    S=4 ELSE 670
650 FOR I=1 TO 16:FOR J=1 TO 16:LOCATE ABS(J+S)-1,
    ABS(K+I)-1:PRINT CHR$(SP(I,J)):NEXT J,I:GOTO 6
    90
660 '** ROTACION **
670 FOR J=20 TO 5 STEP -1:FOR I=5 TO 20: LOCATE J-
    1,I-1:PRINT CHR$(SP(21-J,I-4)):NEXT I,J:GOTO 6
    90
680 '** REPONER CURSOR **
690 C=VPEEK(6145+LX*32+LY):LOCATE LY,LX:PRINT CHR$
    (C+2)
700 FOR I=6 TO 16 STEP 2:LOCATE T,I:PRINT STRING$(
    9,32):NEXT:RETURN 240
710 '** INVERSO **
720 FOR I=0 TO 15:FOR J=0 TO 15:H=VPEEK(6145+(I+4)
    *32+J+4):IF H=100 THEN H=101 ELSE H=100
730 LOCATE J+4,I+4:PRINT CHR$(H):NEXT J,I:GOTO 690
```

```
740 '** MUEVE CURSOR **
750 IF LX+X=20 THEN X=-15 ELSE IF LX+X=3 THEN X=15

760 IF LY+Y=20 THEN Y=-15 ELSE IF LY+Y=3 THEN Y=15

770 LOCATE LY,LX:PRINT CHR$(C):LX=LX+X:LY=LY+Y:LOC
    ATE LY,LX:PRINT CHR$(C+2):GOTO 240
780 '***  OBTIENE SPRITE$ ***
790 LOCATE LY,LX:PRINT CHR$(C);PLAY"S1005AB":M$=""

800 FOR Q=5 TO 13 STEP 8:FOR W=5 TO 13 STEP 8:FOR
    E=0 TO 7:FOR R=0 TO 7:CH=VPEEK(6145+(W+E-1)*32
    +(Q+R-1)):B(E+1,R+1)=CH-100
810 NEXT R,E:GOSUB 960:BEEP:NEXT W,Q
820 M$=LEFT$(M$,LEN(M$)-1):SPRITE$(0)=PT$
830 PT$=""
840 '** GENERA PROGRAMA? **
850 LOCATE T,6:PRINT "G GRABAR":FOR I=8 TO 14 STEP
     2:LOCATE T,I:PRINT STRING$(9,32):NEXT I
860 A$=""
870 LOCATE T,8:A$=INPUT$(1):IF A$<>"G" THEN 690
880 LOCATE T,6:PRINT "1 MAG 0,1":LOCATE T,8:PRINT
     "2 MAG 2,3"
890 LOCATE T,10:A$=INPUT$(1):IF A$="1" THEN M$=LEF
    T$(M$,23) ELSE IF A$<>"2" THEN 890
900 FOR I=6 TO 14 STEP 2:LOCATE T,I:PRINT STRING$(
    9,32):NEXT I
910 LOCATE 0,20:INPUT "NOMBRE";NA$:LOCATE 0,21:INP
    UT "No.LINEA";LI:IF LI>65529! THEN 910
920 '** GRABA EN FORMATO ASCII **
930 OPEN NA$ FOR OUTPUT AS #1:PRINT #1,STR$(LI)+"
    DATA "+M$:CLOSE#1
940 LOCATE 0,20:PRINT STRING$(29*2,32):GOTO 690
950 '*** SUBRUTINA HEX ***
960 FOR R=1 TO 8
970 LW=VAL("&B"+BIN$(B(R,5))+BIN$(B(R,6))+BIN$(B(R
    ,7))+BIN$(B(R,8)))
980 HI=VAL("&B"+BIN$(B(R,1))+BIN$(B(R,2))+BIN$(B(R
    ,3))+BIN$(B(R,4)))
990 M$=M$+HEX$(HI)+HEX$(LW)+",":PT$=PT$+CHR$(VAL("
    &H"+HEX$(HI)+HEX$(LW)))
1000 NEXT R:RETURN
1010 '*** FIGURA DESDE DATOS ***
1020  RESTORE:FOR I=1 TO 4:Y=12-8*(IMOD2):FORK=1TO
     8:X=4-8*(I>2)
1030 READ A$:SP$=SP$+CHR$(VAL("&H"+A$)):A$=RIGHT$(
     "00000000"+BIN$(VAL("&H"+A$)),8)
```

```
1040 FOR J=1 TO 8:B$=MID$(A$,J,1):LOCATE X,Y:PRINT
     CHR$(100+VAL(B$));:X=X+1:NEXT J:Y=Y+1
1050 NEXT K,I:SPRITE$(0)=SP$
1060 GOTO 690
1070 '*** CAMBIA PATRON CARACTER ***
1080 FOR I=CH*8 TO CH*8+7
1090 IF FK=1 THEN READ A$:VP=VAL("&H"+A$)
1100 VPOKE I,VP:NEXT:RETURN
1110 '*** CAMBIA CURSOR ***
1120 PLAY"O4A16":IF C=101 THEN C=100 ELSE C=101
1130 RETURN 350
```

Juego de acción

Introducción

Como muestra de lo que se puede hacer con su Talent-MSX, hemos decidido incluir este sencillo juego donde deberá utilizar astucia y mente ágil. Se trata de la base PLANET RADAR, situada en la Luna, que debe ser defendida de naves visitantes invisibles con muy malas intenciones.
Para poder derribarlas, Ud. deberá detectarlas con el radar antiplasma y luego atacarlas con el clásico rayo Phaser.
Las naves (una por vez) descienden lentamente a la superficie lunar, pero si alunizan estamos perdidos...

Como jugar. Paso a paso

1. Digite **RUN**, luego pulse **RETURN**. Aparece la opción inicial. Pulse **RETURN** para seguir, **ESC** para finalizar.
2. Aparece la pantalla principal. A la izquierda se encuentra la vista frontal del radar, a la derecha, la vista lateral.
3. Para mover el radar en el cuadrante izquierdo: Cursores izquierdo y derecho (mueven izq.-der. respectivamente). Para el cuadrante derecho: Cursores arriba y abajo (mueven der. - izq. respectivamente).
4. Para enviar el haz de radar, pulse **F1**. Para lanzar el rayo Phaser, digite **F2**.
5. El juego finaliza cuando logra alunizar la nave enemiga. Si le acierta antes de esto, la nave se hace visible, se le otorga puntaje de acuerdo a la altura del juego, y pasa a la siguiente vuelta.

Listado

```
10 REM ...     PLANET RADAR     ...
20 REM ... MANUAL DEL USUARIO ...
30 REM ... TALENT-MSX DPC 200 ...
40 SCREEN 1,,0:WIDTH 32:KEY OFF:DEFINTA-Z:COLOR 15
   ,1,1
50 GOSUB 1030
60 C=6145:CH=97
70 R=1:S=0:T=0:U=100:V=7:W=7
80 Y=5:X=RND(-TIME)*12+2:Z=RND(-TIME)*14+1
90 ON KEY GOSUB 360,490
100 KEY(1)OFF:KEY(2)OFF
110 CLS:LOCATE3,10:PRINT"PULSE [RETURN] PARA COMEN
    ZAR":LOCATE 10,12:PRINT"[ESC] PARA FIN"
120 A$=INKEY$:IF A$=CHR$(27) THEN SCREEN 0:COLOR 1
    5,4,4:CLS:END
130 IF A$<>CHR$(13) THEN 120
140 GOSUB 880
150 LOCATE 23,0:PRINT S:LOCATE 23,1:PRINT H:LOCATE
     23,2:PRINT R
160 KEY(1) ON:KEY(2) ON
170 GOSUB 200:F=0:GOSUB 610:T=T+1
180 T=T+(T=30000)*30000:IF F=1 THEN 60
190 GOTO 150
200 REM ...MUEVE EL RADAR...
210 P=V:Q=W
220 K=STICK(0)
230 IF K1=K THEN 270
240 K1=K
250 V=V-(K=3)+(K=7):W=W-(K=1)+(K=5)
260 V=V+(V=14):V=V-(V=-1):W=W+(W=14):W=W-(W=-1)
270 VPOKEC+P+640,0
280 VPOKE C+P+672,CH+7
290 VPOKE C+Q+656,0
300 VPOKE C+Q+688,CH+7
310 VPOKE C+V+640,CH
320 VPOKE C+V+672,CH+1
330 VPOKE C+W+656,CH
340 VPOKE C+W+688,CH+2
350 RETURN
360 REM...DISPARO DEL RADAR...
370 KEY(1)OFF:KEY(2)OFF
380 FOR B=19 TO 5 STEP -1
390 VPOKE C+V+B*32,CH+8:VPOKE C+W+16+B*32,CH+8
400 IF((X-1=V)+(Z-1=W))*(Y=B) THEN 420
410 GOTO 460
```

```
420 SOUND 0,100:SOUND 7,&B10111110
430 SOUND 8,&B00011111:SOUND 12,5
440 SOUND 13,&B1000
450 FOR A=1 TO 500:NEXT A:BEEP
460 NEXTB
470 FOR D=5 TO 19:VPOKE C+V+D*32,0:VPOKE C+W+16+D*
    32,0:NEXT D
480 KEY(1)ON:KEY(2)ON:RETURN 160
490 REM ....DISPARA PHASER....
500 KEY(1)OFF:KEY(2)OFF
510 SOUND7,&B10011100:SOUND 13,&B1001:SOUND 8,&B00
    011111:SOUND 12,10
520 SOUND 9,&B00011111:SOUND 3,2:SOUND 6,100:SOUND
     10,&B00011111
530 F=0:FOR A=19 TO 5 STEP-1
540  VPOKE C+V+A*32,CH+9:VPOKE C+W+16+A*32,CH+9
550  IF (Y=A)*(X-1=V)*(Z-1=W)THEN F=1
560 SOUND 0,255-A*11:SOUND 2,255-A*11
570 NEXT A:BEEP
580 FOR B=5 TO 19:VPOKE C+V+B*32,0:VPOKE C+W+16+B*
    32,0:NEXT B
590 IF F=1 THEN GOSUB 680
600 KEY(1)ON:KEY(2)ON:RETURN 160
610 REM ....MUEVE OVNI....
620 IF (T MOD U)<>0 THEN RETURN
630 Y=Y+1:A=RND(-TIME)*5-2:X=X+SGN(A)
640 A=RND(-TIME)*5-2:Z=Z+SGN(A)
650 X=X-(X=1)+(X=14):Z=Z-(Z=0)+(Z=15)
660 IF Y=20 THEN GOSUB 770:F=1
670 RETURN
680 REM...ACERTADO...
690 SOUND 6,50
700 SOUND 7,&B10110111:SOUND 12,80 :SOUND 13,&B100
    1:SOUND 8,31
710 FOR A=1 TO 50
720  LOCATE X-1,Y:PRINT "def":LOCATE Z+16,Y:PRINT
    "g"
730  LOCATE X-1,Y:PRINT "   ":LOCATE Z+16,Y:PRINT
    " "
740 NEXT A
750 S=S+R*10:R=R+1:Y=5:X=RND(-TIME)*12+2:Z=RND(-TI
    ME)*14+1:U=101-T MOD 43
760 RETURN
770 REM ...FIN JUEGO...
780 KEY(1)OFF:KEY(2)OFF
790 LOCATE X,Y:PRINT "def":LOCATE Z+16,Y:PRINT "g"
800 SOUND 7,56:SOUND 0,0:SOUND 1,0:SOUND 8,8
```

```
810 FOR I=1 TO 15:FOR A=255 TO 0 STEP -4
820 SOUND 0,A
830 NEXT A,I
840 LOCATE 8 ,11:PRINT "FIN DEL JUEGO"
850 IF S>H THEN H=S:LOCATE 22,1:PRINT H
860 PLAY"t160l8o5s1eeffgee4gffDDDF4eeffgee4egegedc
    4"
870 IF PLAY(1) THEN 870 ELSE RETURN
880 REM...PANTALLA INICIAL...
890 CLS
900 LOCATE 1,1:PRINT "PLANET RADAR"
910 LOCATE 16,0:PRINT "PUNTOS="
920 LOCATE 16,1:PRINT "TOPE  ="
930 LOCATE 16,2:PRINT "NIVEL ="
940 C1$=CHR$(215):C2$=CHR$(219)
950 FOR A=4 TO 22
960 LOCATE 0,A:PRINT C1$:LOCATE 15,A:PRINT C1$
970 LOCATE 16,A:PRINT C2$:LOCATE 31,A:PRINT C2$;
980 NEXT A:FOR A=1 TO 14
990 LOCATE A,4:PRINT C1$:LOCATE A,22:PRINT C1$
1000 LOCATE A+16,4:PRINT C2$:LOCATE A+16,22:PRINT
     C2$
1010 LOCATE 1,21:PRINT STRING$(14,104):LOCATE 17,2
     1:PRINT STRING$(14,104)
1020 NEXT A:RETURN
1030 FOR I=776 TO 855
1040 READ A$:A$="&H"+A$:VPOKEI,VAL(A$)
1050 NEXT I
1060 VPOKE 8192+26,&H40:VPOKE 8192+27,&H80:VPOKE 8
     192+6,&H70:VPOKE 8192+7,&H70
1070 FOR I=8192+8 TO 8192+11:VPOKE I,&H30:NEXT:RET
     URN
1080 DATA 10,10,FE,FE,7C,38,10,10,38,38,7C,7C,C6,F
     E,C6,FE
1090 DATA 38,38,38,7C,EE,6C,EE,6C,00,00,01,01,02,0
     4,88,FE
1100 DATA 7C,FE,93,93,D6,7C,38,10,00,00,00,00,80,4
     0,22,FE
1110 DATA 38,7C,E4,E4,74,3C,11,7F,10,38,6C,FE,B7,F
     D,AF,FF
1120 DATA 7C,00,FE,00,7C,00,38,00,10,00,10,00,10,0
     0,10,00
```

Apéndices

Código (decimal)	Código (Hex)	Función	Tecla Equivalente
15	ØF		CTRL + O
16	1Ø		CTRL + P
17	11		CTRL + Q
18	12	Activa o desactiva el modo inserción	CTRL + R , INS
19	13		CTRL + S
2Ø	14		CTRL + T
21	15	Borra la línea actual	CTRL + U
22	16		CTRL + V
23	17		CTRL + W
24	18		CTRL + X , SELECT
25	19		CTRL + Y
26	1A		CTRL + Z
27	1B		CTRL + ! , ESC
28	1C	Mueve el cursor a la derecha	CTRL + / , →
29	1D	Mueve el cursor a la izquierda	CTRL + ! , ←
3Ø	1E	Mueve el cursor hacia arriba	SHIFT + CTRL + ^ , ↑
31	1F	Mueve el cursor hacia abajo	SHIFT + CTRL + - , ↓
127	7F	Borra el caracter donde está cursor	DEL

Apéndice B

TABLA DE CODIGO DE CARACTERES

El código de caracteres disponibles varía desde hex ØØ hasta hex FF (Ø a 255 en decimal). Cada código de caracter dispone de un byte el cual se compone de la parte más significativa y la menos significativa, cada cuatro bits representan un valor numérico.

Letra A mayúscula... Hex 41 (&H41)
Los números hexadecimales llevan la notación del MSX BASIC.

Decimal vs. hexadecimal

Hexadecimal	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	Ø	1	2	3	4	5	6	7	8	9	1Ø	11	12	13	14	15

Los dos dígitos del sistema hexadecimal pueden convertirse en sistema decimal con la siguiente fórmula:

(MSD en base decimal)*16 + (LSD en base decimal)

Ejemplo:

&HFF = 15 * 16 + 15 = 255 (hex FF es 255 en decimal)
&H3B = 3 * 16 + 11 = 59 (hex 3B es 59 en decimal)

MSD: most significant digit , dígito más significativo.
LSD: less significant digit, dígito menos significativo.

Lista de código de caracteres.

Cuando van precedidos por el caracter de "cabecera" gráfico

Los cuatro bits más significativos →

Los cuatro bits menos significativos ↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				Ø	●	P	`	p	C	É	á	Ã			α	≡
1		.	!	1	A	Q	a	q	ü	ee	í	ã			β	±
2		INS	"	2	B	R	b	r	é	Æ	ó	Ĩ			Γ	≥
3			#	3	C	S	c	s	â	ô	ú	ĩ			Π	≤
4			$	4	D	T	d	t	ä	ö	ñ	Õ			Σ	∫
5			%	5	E	U	e	u	à	ò	Ñ	õ			σ	∫
6			&	6	F	V	f	v	å	û	a	Ũ			μ	÷
7	BL		´	7	G	W	g	w	ç	ù	º	ũ			Υ	≈
8	BS		(	8	H	X	h	x	ê	ÿ	¿	Π			Φ	°
9	TAB		)	9	I	Y	i	y	ë	Ö	⌐	ij			Θ	•
A	LF		*	:	J	Z	j	z	è	Ü	¬	¾			ω	Ω	•
B	HOME	ESC	+	;	K	[	k	{	ï	¢	½	~			δ	√
C	CLS	→	,	<	L	\	l	¦	î	£	¼	◊			∞	η
D	CR	←	–	=	M	]	m	}	ì	¥	¡	‰			φ	²
E		↑	.	>	N	^	n	~	Ä	Pt	<	¶T			∈	▮
F		↓	/	?	O	_	o		Å	ƒ	>	§			∩	

Números Hexadecimales ↑

	4	5
0		
1	☺	
2	☻	
3	♥	
4	♦	
5	♣	
6	♠	
7	•	
8	◘	
9	○	
A	◙	
B	♂	
C	♀	
D	♪	
E	♫	
F	☼	

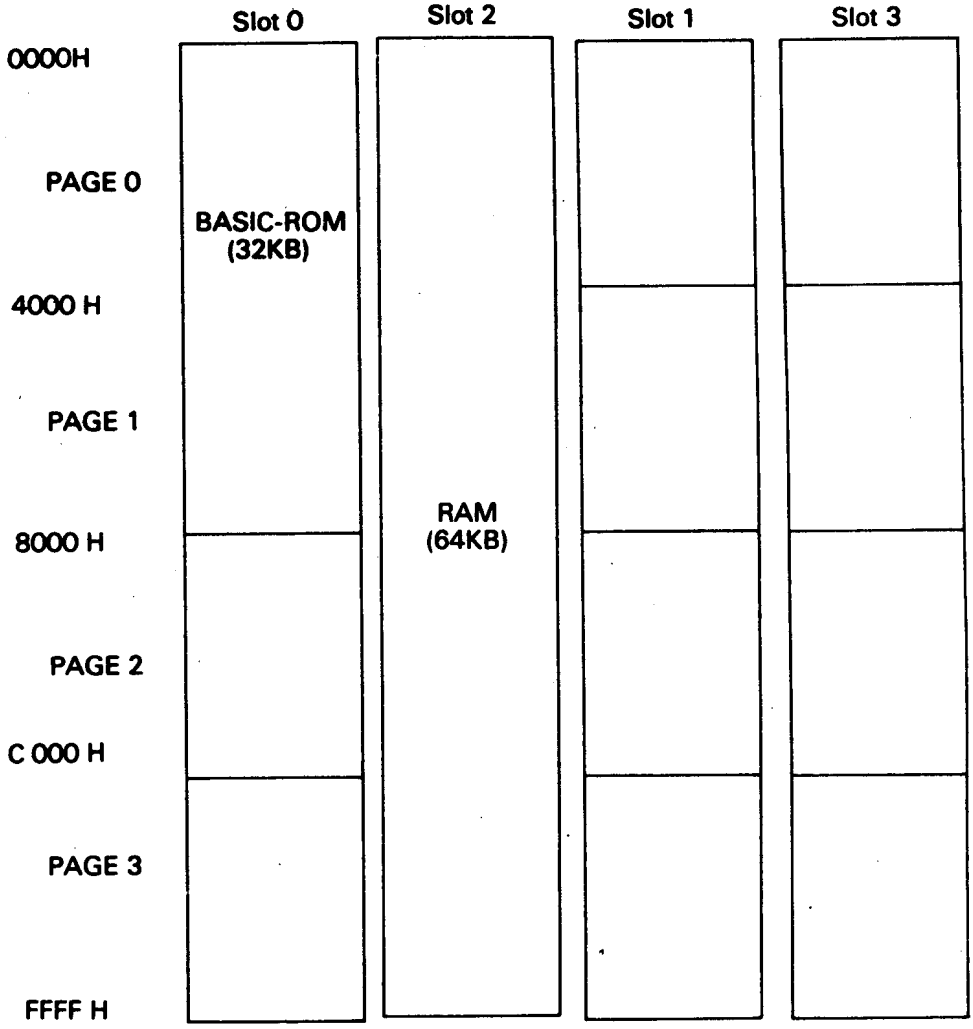
↑ Números hexadecimales

Códigos de control

Los códigos de caracteres con los 4 bits más significativos en Ø ó 1 se generan cuando se pulsa la tecla CTRL y la correspondiente tecla, y se usan varios modos de control.
Ver Apéndice A para el listado completo.

Apéndice C

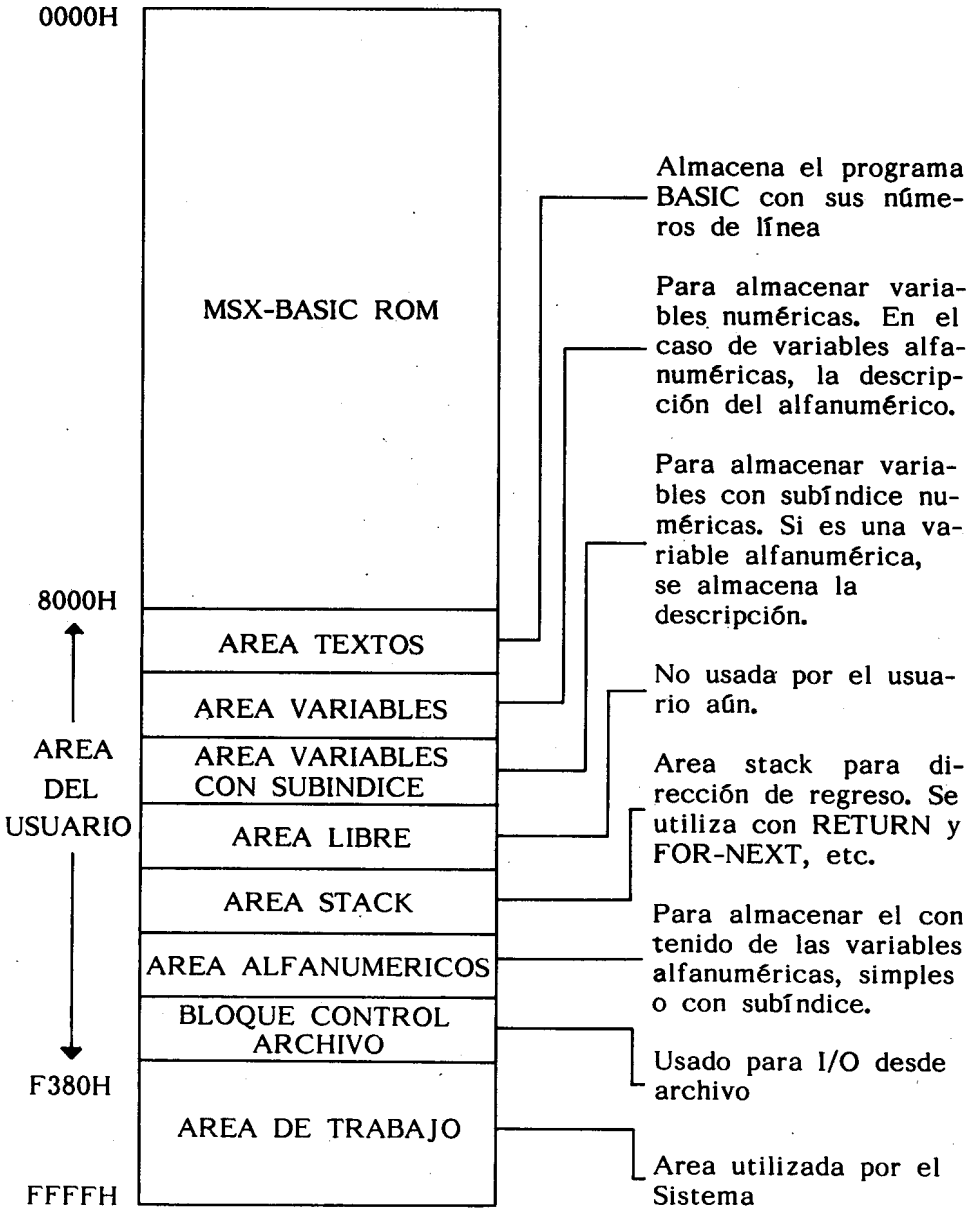
DISTRIBUCION DE SLOTS.



PAGE: Página

Apéndice D

MAPA DE MEMORIA



Apéndice E

MAPA DE I/O

00H	Indefinido
80H	*RS-232C
88H	Reservado por Sistema
90H	IMPRESORA
98H	VDP
A0H	PSG
A8H	PPI
B0H	Reservado por Sistema
D0H	*FDC
D8H	
E0H	Reservado por Sistema
FFH	

dirección I/O	L/E	Descripcion	Comentario
90H	E	Envia señal b0 (strobe)	Señal de "latch"
	L	Lee señal b1 (status)	Se activa por "busy"
91H	E	Salida del dato a imprimir	Señal de "latch"
98H	E	Escribir datos al V-RAM	equivalente a 9929
	L	Leer datos del V-RAM	
99H	E	Envía comando o selección direc-ción memoria	
	L	Lee estado	
A0H	E	Fija dirección memoria (latch)	equivalente a 9129
A1H	E	Envía datos al PSG	
A2H	L	Lee datos del PSG	
A8H	E	Envía datos a la puerta A	equivalente a 8255A
	L	Lee datos de la puerta A	
A9H	E	Envía datos a la puerta B	
	L	Lee datos de la puerta B	
AAH	E	Envía datos a la puerta C	
	L	Lee datos de la puerta C	
ABH	E	Selecciona modo	

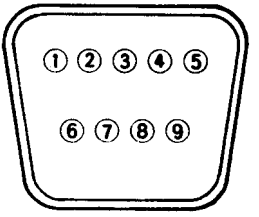
*: significa dispositivo opcional. E=escritura. L=lectura

Apéndice F

CONEXIONES PARA DISPOSITIVOS DE ENTRADA/SALIDA (I/O)

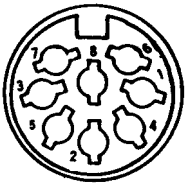
JOYSTICK

PIN	SEÑAL
1	ADELANTE
2	ATRAS
3	IZQUIERDA
4	DERECHA
5	+5V
6	DISPARADOR 1
7	DISPARADOR 2
8	SALIDA
9	TIERRA



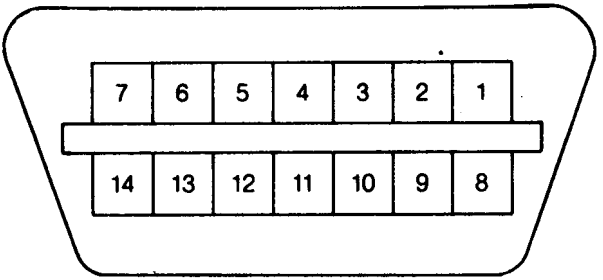
INTERFACE GRABADOR A CASSETTE

PIN	SEÑAL
1	TIERRA
2	TIERRA
3	TIERRA
4	SALIDA CASSETTE
5	ENTRADA CASSETTE
6	REMOTO +
7	REMOTO
8	TIERRA



INTERFACE IMPRESORA

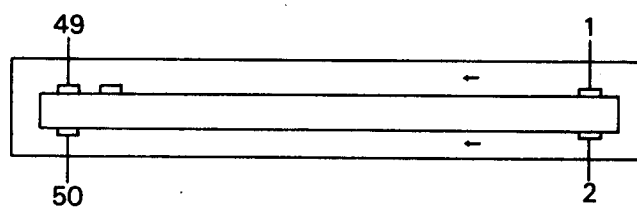
PIN	Tipo	Descripción
1	SALIDA	PSTB
2	SALIDA	PDB0
3	SALIDA	PDB1
4	SALIDA	PDB2
5	SALIDA	PDB3
6	SALIDA	PDB4
7	SALIDA	PDB5
8	SALIDA	PDB6
9	SALIDA	PDB7
10	-	
11	ENTRADA	BUSY
12	-	
13	-	
14	-	TIERRA



CONECTOR DE CARTUCHOS

PIN No.	NOMBRE	E/S	PIN No.	NOMBRE	E/S
1	$\overline{CS1}$	S	26	A13	S
2	$\overline{CS2}$	S	27	A1	S
3	$\overline{CS12}$	S	28	A0	S
4	$\overline{SLTSL}$	S	29	A3	S
5	Reservado *1	-	30	A2	S
6	$\overline{RFSH}$	S	31	A5	S
7	$\overline{WAIT}$ *2	E	32	A4	S
8	$\overline{INT}$ *2	E	33	D1	E/S
9	$\overline{MI}$	S	34	D0	E/S
10	$\overline{BUSDIR}$	E	35	D3	E/S
11	$\overline{IORQ}$	S	36	D2	E/S
12	$\overline{MERQ}$	S	37	D5	E/S
13	$\overline{WR}$	S	38	D4	E/S
14	$\overline{RD}$	S	39	D7	E/S
15	$\overline{RESET}$	S	40	D6	E/S
16	Reservado	-	41	GND	-
17	A9	S	42	CLOCK	S
18	A15	S	43	GND	-
19	A11	S	44	SW1	-
20	A10	S	45	+5V	-
21	A7	S	46	SW2	-
22	A6	S	47	+5V	-
23	A12	S	48	+12V	-
24	A8	S	49	SUNDIN	E
25	A14	S	50	-12V	-

*: No se puede utilizar
*: Open collector



PIN No.	NOMBRE	Descripción
1	CS1	Selecciona señal para dirección 4000H-7FFFH del ROM
2	CS2	Selecciona señal para dirección 8000H-BFFFH del ROM
3	CS12	Selecciona señal para dirección 4000H-BFFFH del ROM
4	SLTSL	Selecciona señal para slot
5	Reservado	Bus reservado para futuras expansiones
6	RFSH	Señal de refresco para RAM dinámicas
7	WAIT	Espera requerida del CPU
8	INT	Interrupción requerida del CPU
9	M1	Ciclo de búsqueda de instrucción del CPU
10	BUSDIR	Señal de control de dirección del buffer del bus de datos externo
11	IORQ	Señal de requerimiento de E/S del CPU
12	MERQ	Señal de requerimiento de memoria del CPU
13	WR	Señal de escritura del CPU
14	RD	Señal de lectura del CPU
15	RESET	Reset del sistema
16	Reservado	Bus reservado para futuras expansiones
17-32	A0-A15	Bus de direcciones
33-40	D0-D7	Bus de datos
41	GND	Tierra
42	Clock	Señal de reloj para el CPU
43	GND	Tierra
44,46	SW1,SW2	Interruptor de protección
45,47	+5V	Línea de alimentación. +5V
48	+12V	Línea de alimentación. +12V
49	SUNDIN	Ingreso de sonido externo
50	-12V	Línea de alimentación. -12V

Apéndice G

LISTA DE CODIGOS DE ERROR POR ORDEN ALFABETICO

Mensaje de error	Cód.	Descripción
Bad file name (Nombre archivo incorrecto)	56	Nombre de archivo incorrecto. * Archivo no abierto con **OPEN** o modo mal especificado
Bad file number (Número archivo incorrecto)	52	Número de archivo incorrecto. * Se especificó un número de archivo mayor que el indicado con **MAXFILES** * Se especificó un número inexistente de archivo (no fue abierto)
Can't continue (No puedo seguir)	17	La ejecución del programa no puede continuar. * Intentó continuar luego de una interrupción por error * Intentó continuar luego de modificar el programa
Device I/O error (Error en dispositivo de entrada/salida)	19	Error en transferencia de datos durante una comunicación con un dispositivo de entrada/salida (I/O)
Direct statement in file (comando directo en archivo)	57	Un archivo ASCII empezado a cargar conteniendo un comando (sentencia directa)
Division by zero (División por cero)	11	Dividido por cero * El divisor era cero * El divisor es una variable indefinida
File already open (Archivo ya abierto)	54	El archivo ya está abierto
File not found (Archivo no hallado)	53	El archivo no ha sido hallado

Mensaje de error	Cód.	Descripción
File not open (Archivo no abierto)	59	El archivo no ha sido abierto
Illegal direct (Comando ilegal)	12	Se intentó ejecutar una sentencia como comando
Illegal function call (Llamada a función ilegal)	5	La sentencia o función es ilegal * El argumento en la sentencia o función excede el rango aceptado * Subíndice de matrices negativo
Internal error (Error interno)	51	Existe un error en el BASIC * Normalmente este tipo de error no ocurre. Si ocurre apagar el equipo durante un tiempo y luego conectar.
Input past end (Ingreso luego de fin de archivo)	55	Se intentó introducir un dato utilizando la sentencia INPUT# después de haber terminado de cargar un archivo. * El número de sentencias INPUT# excede al de datos. * Intentó leer datos en un archivo vacío (Se puede evitar usando EOF como control de fin archivo)
Missing operand (Falta operando)	24	Falta un operando necesario * El número de operandos es incorrecto * Se usaron puntos (.) en vez de comas (,) para separar operandos.
NEXT without FOR (NEXT sin FOR)	1	El número de NEXT excede al de FOR. * Un ciclo FOR - NEXT está mezclado con otro.
No RESUME (Sin RESUME)	21	No existe la sentencia RESUME en una rutina de manejo de errores * El control ha sido devuelto desde una rutina de error con un GOTO

Mensaje de error	Cód.	Descripción
Out of DATA (No hay mas datos)	4	No hay datos para ser leídos por la sentencia **READ** * Insuficiente cantidad de datos * Número de línea erróneo para la sentencia **RESTORE** * Se utilizó separadores erróneos en la sentencia **DATA.**
Out of memory (Memoria completa)	7	Capacidad de memoria completa. * El programa es demasiado largo * Demasiadas variables * Matrices demasiado grandes * Demasiados ciclos **FOR - NEXT** o demasiados **RETURN** sin limpiar stack.
Out of string space (Memoria de alfanuméricos completa)	14	El espacio asignado a las variables alfanuméricas está completo. * El espacio asignado en la sentencia **CLEAR** no alcanza.
Overflow (Desborde)	6	Valor numérico excede el rango permitido. * El resultado del cálculo es muy pequeño o muy grande.
Redimensioned array (Matriz Redimensionada)	10	Se intentó redefinir una variable con subíndice. * Una matriz ya dimensionada ha sido redimensionada por una sentencia **DIM.** * Una matriz con subíndice (no mayor de 10) ha sido usada antes de ser dimensionado con la sentencia **DIM.**
RESUME without error (RESUME sin error)	22	Una sentencia **RESUME** ha sido ejecutada fuera de una rutina de manejo de errores.

Mensaje de error	Cód.	Descripción
RETURN without GOSUB (RETURN sin GOSUB)	3	Una sentencia **RETURN** ha sido ejecutada antes de una sentencia **GOSUB** * Se bifurcó a una subrutina con **GOTO** * No se colocó la sentencia **END** o **STOP** al final del programa principal y la ejecución siguió en una subrutina
String formula too complex (Fórmula alfanumérica muy compleja)		Expresión alfanumérica muy compleja.
String too long (Alfanumérico muy largo)	15	Un alfanumérico es demasiado largo. * Se intenta almacenar más de 255 caracteres en una variable alfanumérica.
Subscript out of range (Subíndice fuera de rango)	9	El subíndice de una matriz es superior al rango permitido. * El valor del subíndice es demasiado grande. Una matriz sin DIM tiene un subíndice mayor que 10.
Syntax error (Error de sintaxis)	2	Sintaxis ilegal para el MSX BASIC. * Línea incorrecta debido a un error de tipeo. * Separadores ilegales (coma, punto, etc.) * Los paréntesis no concuerdan. * Nombre de variable que empieza con un caracter alfabético.
Type mismatch (No concuerdan tipo de variables)	13	Tipos de variables no iguales * Intenta la transferencia de caracteres a una variable numérica. * Los argumentos de una función no concuerdan.

Mensaje de error	Cód.	Descripción
Undefined line number (Número de línea inexistente)	8	Número de línea no definido. * Un número de línea especificado en una sentencia **GOTO**, **GOSUB**, **RESTORE**, o **RESUME** no existe.
Undefined user function (Función no definida)	18	No se encontró la función definida por el usuario. * Nombre incorrecto en **DEF FN.** No se ejecuto la sentencia **DEF FN.**
Verify error (Error en verificación)	20	Error en la verificación de un programa * Se encontró diferencias entre el programa en memoria y el almacenado en cassette cuando se ejecutó una sentencia **CLOAD?** * El programa verificado fue grabado con un sistema con distinta capacidad de memoria RAM al que se está verificando. (Este error ocurre aunque el programa sea correcto)

Apéndice H

PALABRAS RESERVADAS DE MSX-BASIC

Los nombres de funciones o sentencias son palabras reservadas por BASIC. Esto significa que no pueden usarse como nombre de variables. Este apéndice hace una lista completa de estas palabras reservadas. Si intenta utilizarlas, se generará un error.

ABS
AND
ASC
ATN
ATTR$
AUTO
BASE
BEEP
BIN$
BLOAD
BSAVE
CALL
CDBL
CHR$
CINT
CIRCLE
CLEAR
CLOAD
CLOSE
CLS
CMD
COLOR
CONT
COPY
COS
CSAVE
CSNG
CSRLIN
CVD
CVI
CVS
DATA
DEF
DEFDBL
DEFINT
DEFSNG
DEFSTR
DELETE
DIM
DRAW
DSKF
DSKI$
DSKO$
ELSE
END
EOF
EQV
ERASE
ERL
ERR
ERROR
EXP
FIELD
FILES
FIX
FN
FOR
FPOS
FRE
GET
GOSUB
GOTO
HEX$
IF
IMP
INKEY$
INP
INPUT
INPUT$
INSTR
INT
INTERVAL
IPL
KEY
KILL
LEFT$
LEN
LET
LFILES
LINE
LIST
LLIST
LOAD
LOC
LOCATE
LOF
LOG
LPOS
LPRINT
MAX
MERGE
MID$
MKI$
MKS$
MOD
MOTOR
NAME
NEW
NEXT
NOT
OCT$
OPEN
OR
OUT
OFF
ON
PAD
PAINT
PDL
PEEK
PLAY
POINT
POKE
POS
PRESET
PRINT
PSET
PUT
READ
REM
RENUM
RESTORE
RESUME
RETURN
RIGHT$
RND
RSET
RUN
SAVE
SCREEN
SET
SGN
SIN
SOUND
SPACE$
SQR
SPC
SPRITE
SPRITE$
STEP
STICK
STOP
STRIG
STR$
STRING$
SWAP
TAB
TAN
THEN
TIME
TO
TROFF
TRON
USING
USR
VAL
VARPTR
VDP
VPEEK
VPOKE
WAIT
WIDTH
XOR

Apéndice I

ESPECIFICACIONES

ESPECIFICACIONES TALENT MSX DPC 200

Item	Características
Microprocesador	Z80A
Frecuencia del reloj	3.58 MHz
Memoria principal	64K RAM
Memoria del sistema	32K ROM con MSX BASIC incluído
Memoria de video	16K RAM
Lenguaje de programación	BASIC expandido de Microsoft
Salida de pantalla	RF/Videoterminales
VDP	TMS 9929A
Modos de pantalla	4 modos
Modos de escritura	40 caracteres x 24 líneas 32 caracteres x 24 líneas
Modo de alta resolución	256 x 192 puntos
Modo multicolor	64 x 48 bloques (4 x 4 puntos por bloque)
Sprites	32 formas definibles.
Color	16 colores
Teclado	
Tipo:	Mecánico, de desplazamiento, con 77 teclas:
Teclas	48 alfanuméricas 25 de control 4 cursores standard (8 direcciones)
Set de caracteres	Europeo (incluído el español)
Salida de sonido	Terminal de audio
PSG	AY-3-8910 8 octavas Triple acorde

Item	Características
Interface para audio cassette	Universal 8 pin DIN p/leer y grabar
Sistema	FSK 1200/2400 baudios
Interface para impresora	CENTRONICS paralelo
Slots	Slot para cartucho y slot para expansión.
Conexión de joystick	Dos standard
Dimensiones	400 x 225 x 74 mm

Apéndice J

ANTES DE LLAMAR AL SERVICIO TECNICO

(1) No enciende.

Descripción de la falla	Pasos a seguir
No enciende	1. Interruptor apagado. * Asegúrese de que el interruptor esté en la posición "ON". 2. Cable no conectado. * Verifique si está enchufado el cable y si éste no está dañado o cortado. 3. Fusible interno quemado. * Lleve el sistema a su distribuidor para el reemplazo del fusible.

(2) Fallas en la pantalla.

Descripción de la falla	Pasos a seguir
Ninguna señal en la pantalla	1. La computadora está apagada. Enciéndala. 2. Comprobar el sistema de conexión. * Comprobar las conexiones. * Comprobar la correcta conexión de los cables en sus conectores. 3. Si su televisor tiene una entrada de video, comprobar si el selector está situado en una posición errónea 4. Comprobar si está especificado el mismo color para el texto y el fondo con la instrucción COLOR. Primero, pulse [CTRL] y [STOP] y luego simultáneamente las teclas [SHIFT] y [F1 F6]
Imágenes incorrectas en pantalla	Comprobar el conexionado.
Interferencias	1. Comprobar la conexión a la antena de TV. 2. No use nunca un mezclador de antena. Cuando utilice la computadora, debe quitar el cable de la antena exterior de su televisor.

La imagen desborda la pantalla o un extremo de la pantalla desaparece	1. Ajustar el control de anchura de su pantalla. 2. Si el número de caracteres por fila especificados excede el valor inicial (29 ó 37 caracteres por fila), las filas pueden desbordar la pantalla. Corregir el número de caracteres especificados.
Pobre calidad de imagen o pérdida de la sincronización	1. Ajustar el brillo, contraste y/o el sintonizador de canal de su televisor. 2. Ajustar el control de sincronismo vertical u horizontal.
Fuerte dispersión del color	1. Ajustar el control de color de su pantalla. 2. Especificar los siguientes colores: COLOR 15,5 o COLOR 15,4 o COLOR 14,4.

(3) Fallas del teclado.

Descripción de la falla	Pasos a seguir
Teclado bloqueado	1. Apretar las teclas [CTRL] y [STOP] simultánea mente 2. Apagar el equipo momentáneamente y en unos segundos encenderlo de nuevo.

(4) Fallas en el cartucho ROM.

Descripción de la falla	Pasos a seguir
No funciona	* Apagar la computadora y volver a colocar el cartucho en su zócalo. El modo de operación cambia dependiendo del tipo de cartucho. Comprobar cuidadosamente las características e instrucciones del cartucho. * Probar un nuevo cartucho.

(5) Fallas del sistema de audio.

Descripción de la falla	Pasos a seguir
No hay sonido	1. Comprobar si su sistema de audio está desenchufado desde la salida. 2. Comprobar si su sistema de audio está apagado. 3. Comprobar la correcta conexión del sistema. * Comprobar si hay algún error en la conexión

	* Comprobar si los cables están bien conectados en sus enchufes. 4. Si su televisor tiene una entrada de video, el selector de entrada puede estar situado en una posición errónea.
Interferencias	Comprobar que el cable de conexión esté firmemente introducido en su conector. El volumen del televisor es muy elevado. Bájelo

(6) Problemas con el grabador de cassettes.

Si las funciones SAVE o LOAD continúan sin funcionar después de haber efectuado las comprobaciones siguientes, utilice nuestro grabador de cassette recomendado o sea, de calidad.

Caso	Comprobación
Control de volumen	Poner el control de volumen en una posición media - alta. * La carga no se efectuará si el control de volumen está situado en una posición demasiado alta o demasiado baja.
Control de tono	Si su grabador posee un control de tono, colóquelo en una posición media
Comprobación de las pilas	Comprobar que las pilas de su grabador no estén agotadas. * Si lo están, cámbielas por unas nuevas. * Si su grabador tiene un adaptador de corriente alterna (220v) es preferible su utilización.
Proceso a seguir	Compruebe los pasos a seguir de nuevo: * Grabación y lectura Después de grabar su programa o datos, desconectar el cable de conexión al grabador y rebobinar el trozo de cinta grabada. Si la grabación ha sido correcta, se deberá oir un pitido en el parlante. * Si su grabador dispone de VLSS o de alta velocidad de rebobinado, asegurarse de que están desconectados. Si el VLSS está conectado, la grabación falla. Si el mando de alta velocidad de lectura está conectado, la lectura falla.

Limpieza del cabezal	Comprobar si la cabeza de grabación de su grabador necesita limpieza. * Si es necesario, limpiarla con limpiador de cabezales.
Cinta	Compruebe su cinta: * Utilice sólo cintas diseñadas para aplicaciones en computación. Si usted tiene únicamente cassettes de audio, seleccione los de bajo ruido (LN) con duración de 30 a 45 minutos. * Asegúrese de que la cinta no tenga dobleces, empalmes o raspaduras, ya que éstos son causa de fallos. * Utilizar un cassette virgen para grabar su programa o datos. * No utilizar los primeros tramos del cassette.
Velocidad de transmisión	Si durante una grabación en alta velocidad de transferencia (BAUD 2400) ésta falla, reducir a menor velocidad. * En general, una velocidad normal de transferencia asegura la grabación de los programas.

(7) Fallas en los joystick.

Descripción de la falla	Pasos a seguir
Joystick inoperante o sin sensibilidad	1. El joystick está conectado en un conector equivocado. 2. Conexión suelta. 3. El joystick utilizado tiene unas características incompatibles con el Talent MSX DPC 200.

Nota

Si después de efectuar todas las comprobaciones arriba descritas su computadora continúa sin funcionar, le rogamos que contacte con el distribuidor autorizado de Telemática S.A. más próximo.

Apéndice K

INDICE DE INSTRUCCIONES POR APLICACION

Función	Descripción	Sentencias	Pág.
Programación	Borra el programa	NEW	137
	Genera número de línea automáticamente	AUTO	61
	Renumera líneas	RENUM	174
	Borra líneas	DELETE	86
	Imprime el listado del programa en pantalla	LIST	126
	Imprime el listado del programa en impresora	LLIST	128
	Ejecuta el programa	RUN	181
	Reinicia ejecución del programa	CONT	77
	Inicia impresión del número de línea que se está ejecutando	TRON	205
	Finaliza impresión del número de línea que se está ejecutando	TROFF	205
Grabador de cassettes	Graba el programa	CSAVE	78
		SAVE	181
	Carga el programa	CLOAD	72
		LOAD	128
	Verifica el programa	CLOAD?	73
	Mezcla programas	MERGE	133
	Graba programas en código de máquina	BSAVE	67
	Carga programas en código de máquina	BLOAD	66
	Controla el motor	MOTOR ON	137
		MOTOR OFF	137
	Especifica velocidad de transferencia	SCREEN	182
	Especifica cantidad de archivos	MAXFILES	132
	Abre un archivo	OPEN	148
	Imprime datos en archivo	PRINT #	168
		PRINT # USING	168

Función	Descripción	Sentencias	Pág.
	Ingresa datos de archivo	INPUT #	114
		LINE INPUT #	126
		INPUT $	115
	Determina si se llegó al fin de archivo	EOF	92
	Cierra el archivo	CLOSE	73
	Determina la dirección de memoria del bloque de control	VARPTR	208
Teclado	Ingresa datos	INPUT	112
		LINE INPUT	125
		INKEY$	109
		INPUT$	115
	Selecciona el "click" de tecla	SCREEN	182
	Define tecla de función	KEY	118
	Lista las teclas de función	KEY LIST	119
	Imprime el contenido de las teclas de función	KEY ON	120
	Borra la impresión anterior	KEY OFF	120
	Controla las interrupciones por tecla de función	KEY(n) ON	120
		KEY(n) OFF	120
		KEY(n) STOP	120
		ON KEY GOSUB	144
	Controla las interrupciones por las teclas CTRL STOP	STOP ON	198
		STOP OFF	198
		STOP STOP	198
		ON STOP GOSUB	146
	Determina qué cursor se pulsó	STICK	196
	Determina si se presionó la barra espaciadora o no	STRIG	199
	Controla las interrupciones por barra espaciadora	STRIG(n) ON	200
		STRIG(n) OFF	200
		STRIG(n) STOP	200
		ON STRIG GOSUB	147
Control pantalla	Selecciona el modo de pantalla	SCREEN	182
	Especifica el color	COLOR	75
	Borra la pantalla	CLS	74
	Escribe en los registros de VDP	VDP	208
	Lee de los registros de VDP	VDP	208
	Determina la dirección de comienzo de las tablas de VRAM	BASE	63

Función	Descripción	Sentencias	Pág.
	Determina contenido de VRAM	VPEEK	210
	Escribe datos en VideoRAM	VPOKE	211
Pantalla modo texto	Imprime datos	PRINT	161
	Imprime datos formateados	PRINT USING	163
	Imprime el programa	LIST	126
	Determina el ancho de renglón (número de caracteres)	WIDTH	212
	Envía espacios	TAB	203
		SPC	192
	Mueve el cursor	LOCATE	129
	Determina la posición vertical (línea) del cursor	CSRLIN	80
	Determina la posición horizontal (columna) del cursor	POS	159
Pantalla modo gráfico	Dibuja círculos y elipses	CIRCLE	70
	Dibuja líneas y rectángulos	LINE	124
	Dibuja gráficos	DRAW	88
	Rellena con color	PAINT	152
	Dibuja puntos	PSET	169
	Cambia color de puntos	PRESET	160
	Determina el código de color de un punto	POINT	157
	Imprime caracteres y números	MAXFILES	132
		OPEN	148
		PRINT #	168
		PRINT # USING	168
		CLOSE	73
Sprites	Define diseño de sprite	SPRITE$	193
	Coloca sprite en pantalla	PUT SPRITE	170
	Controla las interrupciones por coincidencia sprites	SPRITE ON	193
		SPRITE OFF	193
		SPRITE STOP	193
		ON SPRITE GOSUB	145
Impresora	Imprime datos	LPRINT	132
	Imprime datos formateados	LPRINT USING	132
	Imprime listado del programa	LLIST	128
	Envía espacios	TAB	203
		SPC	192

Función	Descripción	Sentencias	Pág.
	Selecciona impresora MSX ú otras	SCREEN	182
	Determina posición cabezal de impresión	LPOS	131
Sonido	Toca música	PLAY	154
	Determina si se está ejecutando música o no.	PLAY(n)	156
	Envía datos a registros PSG	SOUND	186
	Genera un "bip"	BEEP	64
Joystick, etc.	Determina la dirección del joystick	STICK	195
	Determina si se pulsó el disparador o no	STRIG	199
	Controla las interrupciones por disparador joystick	STRIG ON	200
		STRIG OFF	200
		STRIG STOP	200
		ON STRIG GOSUB	147
	Determina condición "touch pad"	PAD	151
	Determina condición "paddle"	PDL	153
Puerta E/S	Envía datos	OUT	149
	Determina valor de la puerta de ingreso	INT	111
	Espera hasta que se lee el valor especificado	WAIT	211
Archivos	Especifica número de archivos	MAXFILES	132
	Abre un archivo	OPEN	148
	Envía datos	PRINT #	168
	Envía datos formateados	PRINT # USING	168
	Lee datos	INPUT #	114
		LINE INPUT #	126
		INPUT $	115
	Determina si se llegó al fin de archivo o no	EOF	92
	Cierra el archivo	CLOSE	73
	Determina la dirección de memoria del bloque de control	VARPTR	208
Bifurcaciones	El programa bifurca hacia la línea especificada	GOTO	104

Función	Descripción	Sentencias	Pág.
	Determina condición	IF	106
	El programa bifurca hacia diferentes líneas según condición	ON GOTO	142
Subrutinas	Ejecuta una subrutina	GOSUB	103
	Ejecuta una subrutina particular dependiendo de condición	ON GOSUB	142
	El programa regresa a la rutina principal	RETURN	178
Ciclos	Repite ejecución el número indicado de veces	FOR NEXT	97
Errores	Genera error intencionalmente	ERROR	94
	Define la línea inicial de una rutina de manejo de errores	ON ERROR GOTO	140
	El programa regresa desde la rutina de manejo de errores	RESUME	177
	Determina la línea donde se generó el error	ERL	93
	Determina el código de error	ERR	93
Detención	Detiene el programa	STOP	197
Fin	Finaliza el programa	END	91
Comentarios	Inserta comentario en programa	REM	174
Manipulación de alfanuméricos	Reemplaza parte de un alfanumérico	MID$	134
	Determina parte de un alfanumérico	LEFT$ MID$ RIGHT$	122 134 179
	Determina la longitud especificada de caracteres espacio	SPACE$	191
	Genera un alfanumérico de caracter repetido	STRING$	201
	Determina la posición de un alfanumérico dentro de otro	INSTR	116
	Determina la longitud de un alfanumérico	LEN	122

Función	Descripción	Sentencias	Pág.
Conversión de tipos de variables	Convierte un número en real real de doble precisión	CDBL	68
	Convierte un número en entero	CINT	69
	Convierte un número en real de simple precisión	CSNG	79
	Determina el código de caracter	ASC	59
	Convierte un alfanumérico en su valor numérico	VAL	207
	Convierte un código de carac- en su correspondiente caracter	CHR$	68
	Convierte número decimal en binario alfanumérico	BIN$	65
	Convierte número decimal en octal alfanumérico	OCT$	140
	Convierte número decimal en hexadecimal alfanumérico	HEX$	105
	Convierte valor numérico a alfanumérico	STR$	199
Operaciones matemáticas	Determina el arco tangente	ATN	60
	Determina el coseno	COS	78
	Determina el seno	SIN	186
	Determina la tangente	TAN	204
	Determina el exponencial	EXP	96
	Determina el logaritmo	LOG	130
	Determina la raíz cuadrada	SQR	195
	Determina el valor absoluto	ABS	59
	Determina la parte entera	FIX	96
	Determina el máximo entero debajo del número indicado	INT	117
	Determina el signo	SGN	185
Variables	Asigna un valor a una variable	LET	123
	Almacena constantes que serán leídas por la sentencia READ	DATA	81
	Lee constantes almacenadas en las sentencias DATA	READ	173
	Especifica qué sentencia DATA debe leer READ	RESTORE	176
	Define la dimensión de matrices	DIM	87
	Borra una matriz	ERASE	92
	Inicializa todas las variables	CLEAR	71
	Intercambia valores entre dos variables	SWAP	202

Función	Descripción	Sentencias	Pág.
	Determina la dirección de memoria donde está almacenada una variable	VARPTR	208
	Define una variable entera	DEFINT	84
	Define una variable simple precisión real	DEFSNG	84
	Define una variable doble precisión real	DEFDBL	84
	Define una variable alfanumérica	DEFSTR	84
Número aleatorio	Determina el valor de un número aleatorio (al azar)	RND	180
Funciones del usuario	Define una función del usuario	DEF FN	82
Memoria	Determina la cantidad de memoria disponible	FRE	102
	Determina el contenido de una dirección especificada.	PEEK	153
	Escribe un dato en la dirección especificada	POKE	159
	Define el tamaño del área de memoria	CLEAR	71
Lenguaje de máquina	Define la dirección de comienzo de una subrutina en código de máquina	DEFUSR	85
	Ejecuta una subrutina en código de máquina	USR	206
Reloj interno	Inicializa el valor del reloj interno	TIME	204
	Determina el valor del reloj interno	TIME	204
	Controla las interrupciones por intervalo de tiempo	INTERVAL ON INTERVAL OFF INTERVAL STOP ON INTERVAL GOSUB	117 117 117 143
Sentencias extendidas	Llama la sentencia extendida	CALL	67

COMPUTADOR PERSONAL TALENT-MSX DPC-200

Permite integrar múltiples funciones en forma ideal. Su objetivo es satisfacer los diversos requerimientos que le plantea al hombre la era de la informática. Cuando Ud. se pregunta con qué computador se lleva la administración de la casa o de la empresa se comunica con un banco de datos, se aprende informática y juegan chicos y grandes, la respuesta es TALENT-MSX.

¿QUE ES EL MSX-BASIC?

BASIC es el lenguaje por intermedio del cual usted puede comunicarse con su computadora. Fue creado en 1965 en el Dartmouth College por dos maestros: Thomas Kurtz y John Kemeny. La idea original fue la de tener un medio sencillo con el cual programar su computadora.
Pero esto trajo aparejado el problema de la incompatibilidad entre las distintas marcas. La industria informática se ha preocupado de este problema para lo cual planteó una solución en base al concepto de estandarización. En esta empresa están comprometidas ASCII, una exitosa editorial, productora y distribuidora de software de Japón y la MICROSOFT CORPORATION. La solución ASCII-MICROSOFT consistió en crear un estándar, en colaboración con los principales fabricantes japoneses con el objetivo de que se lo reconociera internacionalmente como tal. El resultado final de esta tarea de investigación y desarrollo se denominó NORMA MSX.
Las especificaciones de la NORMA MSX incluyen un diseño circuital basado en el microprocesador Z80 A y otros circuitos integrados, como así también un lenguaje estandarizado, el MSX BASIC.
Este es un lenguaje de alto nivel, muy poderoso y versátil diseñado para ofrecer compatibilidad a nivel software para los diferentes sistemas MSX.
El MSX BASIC ha sido desarrollado a partir del MS-BASIC, al que se han agregado funciones de gráficos con manejo de "sprites", sonido e interrupciones sincrónicas cada 1/50 de segundo.
Se logra así que una gran variedad de paquetes de programas escritos en MSX BASIC puedan ser utilizados en su Talent-MSX DPC-200, y le permita intercambiar con sus amigos que posean sistemas MSX de otras marcas.

© MSX: Marca Registrada de MICROSOFT Corp.
Telemática S.A. - 1986 Todos los derechos reservados.

Producido en Argentina por TELEMATICA S.A.

EL MATERIAL EXHIBIDO
ES SOLO PARA USO
EDUCATIVO, NO COMERCIAL

ENCICLOPEDIA DE
REFERENCIA | EDUCATIVA

RETROTECNIA

PARA ENTENDER LA EVOLUCION DE LOS ORDENADORES A TRAVES DEL TIEMPO